Paralel processing dan pipelining

PARALEL PROCESSING

Pemrosesan Paralel adalah komputasi dua atau lebih tugas pada waktu bersamaan dengan tujuan untuk mempersingkat waktu penyelesaian tugas-tugas tersebut dengan cara mengoptimalkan resource pada sistem komputer yang ada untuk mencapai tujuan yang sama. Pemrosesan paralel dapat mempersingkat waktu ekseskusi suatu program dengan cara membagi suatu program menjadi bagian-bagian yang lebih kecil yang dapat dikerjakan pada masing-masing prosesor secara bersamaan.

Tujuan utama dari pemrosesan paralel adalah untuk meningkatkan performa komputasi. Semakin banyak hal yang bisa dilakukan secara bersamaan (dalam waktu yang sama), semakin banyak pekerjaan yang bisa diselesaikan. Analogi yang paling gampang adalah, bila anda dapat merebus air sambil memotong-motong bawang saat anda akan memasak, waktu yang anda butuhkan akan lebih sedikit dibandingkan bila anda mengerjakan hal tersebut secara berurutan (serial). Atau waktu yg anda butuhkan memotong bawang akan lebih sedikit jika anda kerjakan berdua.

Performa dalam pemrosesan paralel diukur dari berapa banyak peningkatan kecepatan (speed up) yang diperoleh dalam menggunakan teknik paralel. Secara informal, bila anda memotong bawang sendirian membutuhkan waktu 1 jam dan dengan bantuan teman, berdua anda bisa melakukannya dalam 1/2 jam maka anda memperoleh peningkatan kecepatan sebanyak 2 kali.

Adapun proses kerja , pemrosesan paralel membagi beban kerja dan mendistribusikannya pada komputer-komputer lain yang terdapat dalam sistem untuk menyelesaikan suatu masalah. Sistem yang akan dibangun akan tidak akan menggunakan komputer yang didesikasikan secara khusus untuk keperluan pemrosesan paralel melainkan menggunakan komputer yang telah ada. Artinya, sistem ini nantinya akan terdiri dari sejumlah komputer dengan spesifikasi berbeda yang akan bekerjasama untuk menyelesaikan suatu masalah. Adapun tipe-tipe Paralelisme adalah sebagai berikut :

1. Result Paralelisme

Result Paralelisme yang sering disebut sebagai Embarrassingly Parallel atau Perfect Paralel adalah tipe paralelisme dimana komputasinya dapat dibagi menjadi beberapa tugas independen yang mempunyai struktur sama. Data struktur suatu tugas dibagi menjadi beberapa bagian yang berstruktur sama. Contoh tugas yang bisa diselesaikan dengan Result Parallelism adalah Simulasi Montecarlo.

2. Specialist Paralelisme

Cara kerja Specialist Parallelisme adalah dengan mengerjakan beberapa tugas secara bersamaan pada prosesor yang berbeda . Setiap komputer mengerjakan tugas tertentu. Contohnya penggunaanya adalah pada simulasi pabrik kimia, satu

prosesor mensimulasikan proses sebelum reaksi kimia, satu prosesor mensimulasikan reaksi pada tahap awal, dan prosesor lainnya mensimulasikan proses penyulingan hasil, dan seterusnya.

3. Agenda Paralelisme

Tipe paralelisme ini mempunyai daftar yang harus dikerjakan oleh sistem komputer . Semua komputer yang terdapat pada sistem dapat mengakses daftar tersebut. Pada Model MW (Manager Worker) terdapat pengelompokan komputer menjadi dua yaitu :
a) Manager : bertugas memulai perhitungan, memonitor kemajuan tugas, melayani permintaan worker. User berkomunikasi dengan sistem komputer melalui komputer yang berfungsi sebagai manager ini.
b) Worker : mengerjakan tugas-tugas yang diberikan oleh manager. Kerja komputer ini dimulai setelah ada perintah dari manager dan diakhiri oleh manager.
Pesan Terdistribusi dan Lingkungan Pemrograman (Distributed Messaging and Programming Enviroment) hubungannya dengan pemrosesan paralel ialah pesan sebagai sesuatu (objek) pada pemrosesan itu sendiri sehingga kita harus mendeskripsikan pesan (message) itu .

Messages adalah urutan bytes yang dikirimkan antar proses. Pengirim dan penerima harus mempunyai kesepakatan mengenai struktur pesan sehingga isi pesan dapat diterjemahkan dengan benar. Pada prinsipnya cara pengiriman pesan adalah sederhana. Proses A mengirim data buffer sebagai sebuah pesan ke proses B. Pada saat bersamaan proses B menunggu datangnya pesan dari proses A. Ketika pesan tersebut maka proses B akan men-copy pesan tersebut di memori lokalnya. Adapun Metode Pengiriman pesan (message passing) terdapat beberapa metode dalam pengiriman pesan yaitu :

a. Synchronous Message Passing

Cara pengiriman menggunakan metode ini ialah pengirim menunggu untuk mengirim pesan sampai penerima siap untuk menerima pesan. Oleh karena itu tidak ada buffering. Selain itu Pengirim tidak bisa mengirim pesan untuk dirinya sendiri.

b. Ansynchronous Message Passing

Pengirim akan mengirim pesan kapanpun dia mau. Pengirim tidak peduli ketika penerima belum siap untuk menerima pesan. Oleh karena itu diperlukan buffering untuk menampung pesan sementara sampai penerima siap menerima pesan. Selain itu pengirim dapat pesan untuk dirinya sendiri. Selain berdasarkan metode pengiriman pesan diatas, pengiriman pesan (message passing) dibedakan berdasarkan jumlah penerima pesan yaitu: Point to Point dan Broadcast. Perbedaan mendasar keduanya adalah jumlah penerima yang menerima pesan. Pada Point to Point penerimanya tunggal sedangkan pada broadcast jumlah penerimanya banyak.

PVM dan MPI Mesin virtual parallel atau Parallel Virtual Machine (PVM) dan Antarmuka Pengiriman Pesan (MPI) adalah kumpulan library yang memungkinkan kita untuk menulis program pengiriman pesan paralel menggunakan bahasa pemrograman C dan FORTRAN agar bisa berjalan pada sistem paralel.

Kemampuan sistem paralel tergantung dari kemampuan pemrogram untuk membuat aplikasi terdistribusi ketika dijalankan pada sistem paralel. Jika node slave mempunyai prosessor lebih dari satu maka pemrogram harus memperhitungkan kemungkinan paralelisme 2 level: Paralelisme di dalam slave node (intra-node parallelism) dan paralelisme antar slave node (inter-node parallelism). Inter-node parallelism menggunakan shared memory dalam node sehingga tidak melakukan pertukaran data secara explicit. Sedangkan Inter-node parallelism melakukan pertukaran data lewat media yang menghubungkan antara node slave yang ada.

Terdapat tiga metode untuk mengimplementasikan Inter-node parallelism yaitu :

a. Dengan cara membuat protokol komunikasi ad hoc level rendah. Contohnya dengan menggunakan socket interface.
b. Dengan menggunakan distributed communication library. Contohnya dengan menggunakan Message Passing Interface (MPI) library
c. Dengan memanfaatkan layer software dengan maksud untuk menyembunyikan interconnect dari programmer.
Setelah kita mengenal , komponen – komponen dan tujuan dari pemrosesan paralel tersebut, kita beralih ke sistem pemrosesan paralel dimana, sistem pemrosesan paralel adalah sekumpulan komputer terhubung dan bekerjasama sebagai satu resource komputer yang terintegrasi untuk menyelesaikan suatu tujuan. Sebuah sistem paralel setidaknya terdiri dari Message Passing Interface (MPI) dan sebuah pengatur beban kerja (job scheduler) . Message Passing Interface bertugas untuk mengirim data antar komputer di dalam sistem paralel (biasanya disebut sebagai node atau host). Job scheduler seperti yang tersirat dari namanya bertugas menerima tugas dari user dan menjadwalkan tugas tersebut pada beberapa node didalam sistem parallel sesuai kebutuhan.
MPI (Message Passing Interface) adalah sebuah mekanisme mengiriman instruksi dan data antara dua proses komputasi yang berbeda yang berada pada komputer berbeda pada sistem sistem paralel. Paket-paket yang mempunyai spesifikasi kebutuhan MPI telah banyak beredar di Internet dan telah dilengkapi dengan LAM/MPI [5] dan MPICH [6]. Paket-paket ini telah dilengkapi dengan fungsi-fungsi yang menggunakan library C dan Fortran. Kemampuan MPI digunakan untuk menginterpretasikan bahasa pemrograman matrik kemampuan dynamic linking dari bahasa tersebut. Fungsi library dari paket MPI dapat digabungkan dengan dynamic extension dengan cara menghubungkan bahasa pemrograman tersebut dengan bahasa C, C++, atau FORTRAN. Hal ini telah dilakukan untuk menciptakan toolbox MPI (MPITB) untuk kebutuhan MATLAB, dan bahasa pemrograman GNU Octave oleh Fernandez Baldomero . Pada makalah ini digunakan MPITB dengan pertimbangan fungsionalitas dan kelengkapannya disamping fakta bahwa MPITB dan GNU Octave adalah bebas digunakan bahkan untuk keperluan komersial. Hal ini juga berarti bahwa source code-nya banyak beredar dan dapat dimodifikasi sesuai kebutuhan.
Aplikasi Pemrosesan Paralel
Desain Jaringan

Secara garis besar, mekanisme pemberian layanan publik bagi user di luar jaringan adalah sebagai berikut:

a) User diluar jaringan diatas melakukan suatu request tugas, misalnya tracetest_example.m
b) Request diterima oleh Load balancer/Linux box untuk kemudian diolah dan dibagi menjadi proses yang relatif lebih kecil
c) Proses yang telah berukuran kecil tersebut diolah oleh masing-masing node/slave untuk diselesaikan.
d) Setelah selesai melakukan tugasnya, node/slave mengirimkan kembali hasilnya ke Load balancer untuk kemudian disusun kembali.
e) Hasilnya dikirimkan kembali ke user.

Untuk menerangkan bagaimana jaringan sistem paralel bekerja , kami mengambil contoh dari salah satu penelitian kecil , dalam penelitian ini hanya akan digunakan lima buah komputer karena adanya keterbatasan resource yang ada. Sistem sistem paralel ini tidak terhubung ke Internet. Diasumsikan bahwa user menggunakan komputer Server/Master. Jikapun ada user yang berada dalam jaringan internet menginginkan untuk menggunakan sistem sistem paralel ini, maka ia dapat mengakses komputer server/master menggunakan program aplikasi Telnet/ssh.

Kesimpulan
Pemrosesan paralel dipergunakan untuk memudahkan user dalam berinteraksi dari satu sistem ke sistem yang lain, dengan tujuan untuk membagi beban yang terdapat pada suatu sistem sehingga satu masalah dipecahkan secara bersama-sama. Dan keberhasilan pemrosesan paralel itu dapat dilihat dari kecepatan (speed up) yang diperoleh dari teknik paralel yang digunakan. Tipe – tipe paralelisme terbagi tiga yaitu : (1) Result Paralelisme, (2) Specialist Paralelisme , dan (3) Agenda Paralelisme. Dari pemrosesan paralel itu , tentunya ada suatu tindakan dari proses tersebut yang dikenal sebagai sistem pemrosesan paralel. Kemudian, aplikasi dari pemrosesan paralel itu salah satunya untuk desain jaringan.

PIPELINING

Pipeline adalah suatu cara yang digunakan untuk melakukan sejumlah kerja secara bersama tetapi dalam tahap yang berbeda yang dialirkan secara kontinu pada unit pemrosesan. Dengan cara ini, maka unit pemrosesan selalu bekerja.

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

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.

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 X kali dibandingkan dengan microprocessor yang tidak menggunakan pipeline, apabila tahapan yang ada dalam satu kali pemrosesan instruksi adalah X tahap.

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.

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 microprocessoryang 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.

2. Kenapa komputer menggunakan teknik Pipelining??

- Drive for computing speed never ends.
- Improvements from architecture or organization point of view are limited
- Clock speed enhancement is done, but more improvement should be sought from
- instruction execution perspective, instead of hardware design
- Flyn’s Taxonomy : SISD (Single Instruction Single stream of Data), SIMD (Single
- Instruction Multiple stream of data) or MIMD – Parallel
- Parallel Processor : may be a solution
- Use two processors (or more, instead of one) in a computer system
- How do it runs the code ? (program)
- Suppose, we have a problem :

C = (A2+ B2)

3. Intruksi pipeline

Tahapan pipeline

- Mengambil instruksi dan membuffferkannya
- Ketika tahapn kedua bebas tahapan pertama mengirimkan instruksi yang dibufferkan tersebut
- Pada saat tahapan kedua sedang mengeksekusi instruksi, tahapan pertama memanfaatkan siklus memori yang tidak dipakai untuk mengambil dan membuffferkan instruksi berikutnya .

Instuksi pipeline:

Karena untuk setiap tahap pengerjaan instruksi, komponen yang bekerja berbeda, maka dimungkinkan untuk mengisi kekosongan kerja di komponen tersebut. Sebagai contoh :

Instruksi 1: ADD AX, AX

Instruksi 2: ADD EX, CX

Setelah CU menjemput instruksi 1 dari memori (IF), CU akan menerjemahkan instruksi tersebut(ID). Pada menerjemahkan instruksi 1 tersebut, komponen IF tidak bekerja. Adanya teknologi pipeline menyebabkan IF akan menjemput instruksi 2 pada saat ID menerjemahkan instruksi 1. Demikian seterusnya pada saat CU menjalankan instruksi 1 (EX), instruksi 2 diterjemahkan (ID).

Contoh pengerjaan instruksi **tanpa** pipeline

t =	1	2	3	4	5	6	7	8	9	10
ADD AX,AX	IF	DE	IF	DE	EX					
ADD BX,CX						IF	DE	IF	DE	EX

Disini instruksi baru akan dijemput jika instruksi sebelumnya telah selesai dilaksanakan.

t =	1	2	3	4	5	6	7	8	9	10
ADD AX,AX	IF	DE	IF	DE	EX					
ADD BX,CX		IF	DE	IF	DE	EX				
ADD DX,DX			IF	DE	IF	DE	EX			

Contoh pengerjaan instruksi dengan pipeline

Disini instruksi baru akan dipanggil setelah tahap IF menganggur (t2).

Dengan adanya pipeline dua instruksi selesai dilaksanakan pada detik keenam (sedangkan pada kasus tanpa pipeline baru selesai pada detik kesepuluh). Dengan demikian telah terjadi percepatan sebanyak 1,67x dari 10T menjadi hanya 6T. Sedangkan untuk pengerjaan 3 buah instruksi terjadi percepatan sebanyak 2, 14 dari 15T menjadi hanya 7T.

Untuk kasus pipeline sendiri, 2 instruksi dapat dikerjakan dalam 6T(CPI = 3) dan instruksi dapat dikerjakan dalam 7T (CPT = 2,3) dan untuk 4 instruksi dapat dikerjakan dalam 8T (CPI =2). Ini berarti utnuk 100 instruksi akan dapat dikerjakan dalam 104T (CPI = 1,04). Pada kondisi ideal CPI akan harga 1.

4. Permasalahan di (dalam) Instruksi Pipelining

- VARIASI WAKTU:

Tidak semua tahap memakan waktu yang sama. Ini berarti untuk mendapatkan kecepatan dalam intruksi pipelining sangat ditentukan oleh tahap yang paling lambat. Masalah ini sangat akut dalam memproses instruksi, sejak instruksi yang berbeda memiliki persyaratan

operand waktu proses yang berbeda. Selain itu, diperlukan mekanisme sinkronisasi untuk memastikan bahwa data lewat dari stage ke stage hanya ketika kedua stage siap.

- DATA BERBAHAYA (DATA HAZARDS):

Ketika beberapa instruksi di eksekusi secara parsial, masalah timbul jika mereka referensi data yang sama. Kita harus memastikan bahwa instruksi selanjutnya tidak berusaha untuk mengakses data lebih cepat dari instruksi sebelumnya, jika ini terjadi akan menyebabkan hasil yang salah. Sebagai contoh, instruksi N +1 tidak harus diperbolehkan untuk mengambil sebuah operand yang belum disimpan oleh instruksi N.

- PERCABANGAN (BRANCH):

untuk mengambil instruksi berikutnya, kita harus tahu mana saja yang dibutuhkan, Jika instruksi ini adalah cabang bersyarat (conditional branch) instruksi berikutnya mungkin tidak diketahui sampai saat diproses.

- JEDA(INTERUPTSI):

interupsi membuat instruksi extra yang tidak terencana untuk masuk kedalam aliran intruksi. jeda(Interrupt) harus berperan antar instruksi. yaitu, ketika satu instruksi telah selesai dan berikutnya belum dimulai. Dengan pipelining, instruksi berikutnya biasanya dimulai sebelum yang sekarang telah selesai.

Semua masalah ini harus diselesaikan dalam konteks kebutuhan kita untuk mendaatkan kinerja dengan kecepatan tinggi. Jika kita tidak dapat mencapai kecepatan yang cukup, pipelining mungkin tidak sepadan.

5. Beberapa Solusi

Strategi pemecahan masalah diatas adalah sebagai berikut :

- VARIASI PEMILIHAN WAKTU (TIMING VARIATIONS)

Untuk memaksimalkan kecepatan, untuk menyeragam, tahap pertama yang harus dilakukan adalah mekanisme waktu yang diperlukan. Sebuah metode sinkron dapat digunakan,jika tahapan telah dianggap lengkap dari sejumlah tertentu siklus waktu. Namun, teknik asynchronous secara umum lebih efisien. Bit atau garis sinyal dilewatkan maju ke tahap berikutnya menunjukkan data sudah valid. Sebuah sinyal juga harus lulus kembali dari tahap berikutnya ketika data telah diterima.

Dalam semua kasus harus ada sebuah penyangga antar tahap untuk menyimpan data, kadang-kadang penyangga ini diperluas ke memori yang dapat menyimpan beberapa item data. Setiap tahap harus berhati-hati untuk tidak menerima input data sampai berlaku, dan tidak untuk menghasilkan data output sampai ada ruang dalam penyangga (buffer).

- DATA BERBAHAYA (DATA HAZARDS)

Untuk melindungi dari data yang berbahaya, perlu untuk menyadari setiap tahap yang digunakan oleh tahap pipelining yang lebih jauh . Jenis penggunaan juga harus diketahui, dua pembacaan yang terurut tidak boleh bertentangan dan tidak boleh menyebabkan perlambatan pada pipeline. Namun akan ada kemungkinan konflik jika terjadi penulisan.

Pipeline biasanya dilengkapi dengan small associative check memory yang dapat menyimpan alamat dan jenis operasi (read atau write) untuk setiap instruksi yang ada di pipeline. Konsep “alamat(address)t” harus diperluas untuk mengidentifikasi register. Setiap instruksi hanya dapat mempengaruhi sejumlah kecil dari operand, tapi efek yang tidak langsung addressing tidak boleh diabaikan.

Ketika instruksi bersiap memasuki pipa, alamat operan telah disimpan.Jika ada konflik, instruksi (dan yang di belakangnya) harus menunggu. Ketika ada konflik, instruksi memasuki pipa dan alamat operan disimpan dalam memori cek. Ketika instruksi selesai, alamat ini akan dihapus. memori harus bisa untuk menangani proses pencarian berkecepatan tinggi yangdiperlukan.

- PERCABANGAN (BRANCHING)

Masalah dalam percabangan adalah pipelining diperlambat oleh instruksi karena kita tidak tahu cabang yang mana yang harus kita ikuti. Dengan tidak adanya bantuan spesial dalam masalah ini, perlu menunda pemrosesan instruksi selanjutnya sampai tujuan percabangan diselesaikan. Karena cabang sangat sering, penundaan ini bersifat tidak dapat diterima.

Salah satu solusi yang banyak digunakan, terutama di RISC, adalah menunda percabangan. Dalam metode ini, rangkaian instruksi ini dirangkai sedemikian rupa, sehingga setelah suatuinstruksi, instruksi berikutnya bisa selalu di ekskusi, dan kemudian cabang di hapus. Dengan begitu tiap-tiap cabang harus diikuti oleh satu instruksi yang secara logika mendahuluinya dan diharapkan untuk dieksekusi dalam semua kasus. Hal ini akan memberikan ruang bernafas (istirahat) bagi pipeline. Jika perlu, instruksi ini bisa merupakan suatu no-op, tapi, seringnya penggunaan no-op akan merusak fungsi kecepatan.

Penggunaan teknik ini memerlukan metode pengkodean yang membingungkan bagi progammer, tetapi tidak terlalu sulit bagi generator kode compiler.

Kebanyakan teknik lain menggunakan ekskusi spekulatif, di mana instruksi yang diproses yang tidak dikenal dengan pasti, dianggap sebagai benar. Hal ini harus dihindari, harus dibuang dan tidak di simpan.

Solusi yang biasa digunakan adalah dengan mengikuti cabang yang jelas, yaitu instruksi sekuensial berikutnya, berhati-hati untuk tidak melakukan tindakan yang tidak bisa dirubah. Operan mungkin diambil dan di proses, tetapi tidak akan ada hasil sampai cabang di terjemahkan. Jika pilihan itu salah, dapat di tinggalkan dan cabang alternatif dapat di proses.

Metode ini bekerja dengan cukup baik jika cabang jelas dan benar. Ketika coding menggunakan pipelined CPU, perawatan harus dilakukan untuk kode cabang (terutama transfer error) sehingga jalur luruslah yang biasanya diambil. Tentu saja, bercabang yang tidak di perlukan harus dihindari.

Kemungkinan lain adalah untuk menyusun kembali program sehingga cabang ada lebih sedikit, misalnya dengan tidak mengikuti jenis loop tertentu. Ini dapat dilakukan dengan mengoptimalkan kompiler atau, dalam beberapa kasus, dengan perangkat keras itu sendiri.

Sebuah strategi yang pada umumya di gunakan oleh banyak arsitektur saat ini beberapa jenis prediksi cabang. Hal ini mungkin berdasarkan informasi yang diberikan oleh kompilator atau pada statistik yang dikumpulkan oleh perangkat keras. Tujuaannya adalah membuat perkiraan terbaik apakah cabang tertentu akan diambil atau tidak, dan menggunakan perkiraan ini untuk melanjutkan pipelining.

Solusi yang membutuhkan harga yang lebih, kadang-kadang digunakan untuk memisahkan pipeline dan memulai memproses kedua cabang. Gagasan ini mendapat perhatian baru dalam beberapa prosesor terbaru.

6. Keuntungan dan Kerugian

Pipelining tidak membantu dalam semua kasus. Ada beberapa kemungkinan kerugian. Pipa instruksi dikatakan sepenuhnya pipelined jika dapat menerima instruksi baru setiap clock cycle. Sebuah pipa yang tidak sepenuhnya pipelined telah menunggu siklus yang menunda kemajuan pipa.

Keuntungan dari Pipelining:

1. Waktu siklus prosesor berkurang, sehingga meningkatkan tingkat instruksi dalam kebanyakan kasus(lebih cepat selesai).
2. Beberapa combinational sirkuit seperti penambah atau pengganda dapat dibuat lebih cepat dengan menambahkan lebih banyak sirkuit. Jika pipelining digunakan sebagai pengganti, hal itu dapat menghemat sirkuit & combinational yang lebih kompleks.
3. Pemrosesan dapat dilakukan lebih cepat, dikarenakan beberapa proses dilakukan secara bersamaan dalam satu waktu.

Kekurangan Pipelining:

1. Pipelined prosesor menjalankan beberapa instruksi pada satu waktu. Jika ada beberapa cabang yang mengalami penundaan cabang (penundaan memproses data) dan akibatnya proses yang dilakukan cenderung lebih lama.
2. Instruksi latency di non-pipelined prosesor sedikit lebih rendah daripada dalam pipelined setara. Hal ini disebabkan oleh fakta bahwa intruksi ekstra harus ditambahkan ke jalur data dari prosesor pipeline.
3. Kinerja prosesor di pipeline jauh lebih sulit untuk meramalkan dan dapat bervariasi lebih luas di antara program yang berbeda.

4. 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.
5. Sedangkan ketergantungan terhadap data, bisa muncul, misalnya instruksi yang berurutan memerlukan data dari instruksi yang sebelumnya.
6. 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.

.

7. Kesulitan dalam Pipeline

Untuk menerapkan prinsip *multi-stage* atau mulai saat ini kita namakan pipelining di prosesor, diperlukan organisasi prosesor khusus. Pada dasarnya, prosesor dipartisi menjadi sejumlah unit-unit kecil dengan fungsi spesifik. Setiap unit berperan untuk menyelesaikan sebagian dari instruksi-intruksi berikut :

Instruction fetch, decode, operand address calculation, operand fetch, execute dan store result.

Dalam proses di atas terkadang sering terjadi kendala/conflict seperti:

- Terjadinya pause (P_i), karena adanya data conflict dalam program tersebut
- Terjadinya data error dikarenakan banyaknya proses yang dilakukan bersamaan
- Terjadinya pengambilan data secara bersamaan, sehingga salah satu proses tertunda
- Terjadinya penumpukan data di salah satu intruksi sehingga ada beberapa proses yg di tunda
- Dengan terjadinya conflict tadi, speed-up yang diperoleh menjadi lebih kecil (lambat) dibandingkan dengan tanpa conclict.