Arsitektur dan Organisasi Komputer:
Struktur Prosesor dan Fungsi

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

JOIN UB BE THE BEST
BRAWIJAYA UNIVERSITY

12

SELF-PROPAGATING ENTREPRENEURIAL EDUCATION DEVELOPMENT

1. PENDAHULUAN

1.1 Pengantar

Bagian ini membahas aspek – aspek struktur dan fungsi CPU untuk dasar pembahasan bab berikutnya, yaitu bab 6 RISC. Fokus bab struktur dan fungsi CPU adalah organisasi prosesor dan register, siklus instruksi dan strategi dalam metode pipelining.

1.2 Tujuan

Mengerti struktur dan Fungsi CPU yaitu dapat melakukan Fech Instruksi, Interpreter instuksi, Fech data, exekusi, dan menyimpan kembali. Serta struktur dari register macam-macam register dan fungsinya

Mengerti aliran data pada siklus pengambilan, siklus tak langsung, siklus interupt, Mengerti pipelining, dan mengerti teknik-teknik menangani percabangan pada pipelining

1.3 Latar belakang

Dewasa ini sistem computer multiguna terdiri dari CPU (*central processing unit*) sejumlah *device controller* yang dihubungkan melalui *bus* yang menyediakan akses ke memori . Setiap *device controller* bertugas mengatur perangkat yang spesifik (contohnya *disk drive*,*audio device*, dan *video display*). CPU dan *device controller* dapat dijalankan secara bersamaan, namun demikian diperlukan mekanisme sinkronisasi untuk mengatur akses ke memori.

2. Organisasi CPU

CPU merupakan komponen terpenting dari sistem komputer. CPU adalah komponen pengolah data (fetch data, write & read data) berdasarkan instruksi-instruksi (fetch, interrupt, dll) yang diberikan kepadanya. Dalam mewujudkan fungsi dan tugasnya, CPU tersusun atas beberapa komponen sebagai bagian dari struktur CPU, seperti terlihat pada gambar 2 dan struktur detail internal CPU terlihat pada gamber 3. CPU tersusun atas beberapa komponen, yaitu :

1. *Arithmetic and Logic Unit* (ALU), bertugas membentuk fungsi – fungsi pengolahan data komputer. ALU sering disebut *mesin bahasa* (*machine language*) karena bagian ini mengerjakan instruksi – instruksi bahasa mesin yang diberikan padanya. Seperti istilahnya, ALU terdiri dari dua bagian, yaitu unit arithmetika dan unit logika boolean, yang masing-masing memiliki spesifikasi tugas tersendiri.
2. *Control Unit*, bertugas mengontrol operasi CPU dan secara keselurahan mengontrol komputer sehingga terjadi sinkronisasi kerja antar komponen dalam menjalankan fungsi – fungsi operasinya. Termasuk dalam tanggung jawab unit kontrol adalah mengambil instruksi-instruksi dari memori utama dan menentukan jenis instruksi tersebut.
3. *Registers*, adalah media penyimpan internal CPU yang digunakan saat proses pengolahan data. Memori ini bersifat sementara, biasanya digunakan untuk menyimpan data saat diolah ataupun data untuk pengolahan selanjutnya. Register merupakan memori dengan level hirarki paling tinggi berada di atas memori utama dan cache oleh karena itu register lebih cepat, lebih kecil dan akses data per bit.

   Register pada CPU memiliki 2 fungsi, yaitu:

   1) User visible-Register

      Register ini memungkinkan programmer bahasa mesin dan bahasa assembler meminimalkan referensi main memory dengan cara mengoptimasi penggunaan register. User visible-Register adalah register yang dapat direferensikan dengan menggunakan bahasa mesin yang dieksekusi CPU.

      User visible-Register memiliki kategori:

      a. General purpose

Bersifat terbatas dan biasanya digunakan untuk data atau pengelamatan. General purpose register dapat digunakan untuk berbagai fungsi oleh pemrogram sehingga bisa lebih flexible dan membuat instruksi lebih cepat diprosses. General-purpose register dapat dapat berisi operand sembarang opcode. Pada kasus-kasus tertentu, general-purpose register dapat digunakan untuk fungsi-fungsi pengalamatan (misalnya, register indirect, displacement). Pada kasus lainnya, terdapat partial atau batasan yang jelas antara register data dengan register alamat. General purpose memiliki kapasitas optimum yaitu dengan ukuran antara 8 sampai 32 dan kapasitasnya dapat menyimpan 1 alamat penuh atau 1 kata penuh.

b. Data

Register data hanya dapat dipakai untuk menampung data dan tidak dapat digunakan untuk kalkulasi dan alamat operand.

c. Address

Register alamat menyerupai generalpurpose, atau register-register tersebut dapat digunakan untuk mode pengalamatan tertentu.

d. Condition codes

Condition codes biasanya di-set per bit, condition codes merupakan code dengan kondisinya sudah diperhitungkan pada instruksi program (seperti jump if zero) namun kondisi tersebut tidak dapat diset oleh program.

2) Control & Status Register

Register ini digunakan oleh unit kontrol untuk mengontrol operasi CPU dan oleh program sistem operasi untuk mengontrol eksekusi program. Register ini tidak visible untuk user namun visible terhadap instruksi mesin yang dieksekusi pada mode kontrol atau sistem operasi.

Register yang penting bagi eksekusi instruksi adalah:

a. Program Counter (PC) atau Pencacah Program: berisi alamat instruksi yang akan diambil,
b. Instruction Register (IR): berisi instruksi yang terakhir diambil,
c. Memori Address Register (MAR): berisi alamat sebuah lokasi di dalam memori,
d. Memori Buffer Register (MBR): berisi sebuah word data yang akan dituliskan ke dalam memori atau word yang terakhir dibaca.

Biasanya, CPU updatePCsetelahsetiap instruksifetchsehingga PCselalumenunjuk ke instruksi berikutnya yang akandieksekusi.Sebuah branchatau skip instructionjuga akanmemodifikasi isi dariPC.Instruksiyang diambildimasukkan ke dalamIR, di manaopcodedan operanspecifierdianalisis.Datadipertukarkandenganmemori dengan menggunakanMARdan MBR.Dalamsistem bus-organized,MARterhubung langsung ke address-bus,dan MBR terhubunglangsung kedata bus. User visible-Register,ketika digunakan maka terjadipertukaran data denganMBR.

Keempatregisteryang disebutkandigunakan untukpergerakan dataantara CPUdan memori.Dalam CPU,data harus disajikan ke ALU untuk diproses. ALU dapat memiliki akses langsung ke MBR dan User visible-Register.Atau, mungkin ada register penyangga tambahan di batas ke ALU: register ini berfungsi sebagai input dan output register untuk semua dan pertukaran data dengan MBR dan User visible-Register.

Semuadesain CPU termasuk sebuah register atau set register sering dikenal sebagai Program Status Word (PSW), yang berisi informasi status. PSW biasanya berisi kode kondisi ditambah informasi lainnya. Bidang umum atau flag meliputi:

1) Sign

   Berisi bit tanda hasil operasi aritmetika terakhir, negatif atau positf

2) Zero

   Diset bila hasil sama dengan nol

3) Carry

   Diset apabila operasi yang dihasilkan di dalam carry (penambahan) ke dalam bit yang lebih tinggi atau borrow (pengurangan pengurangan) dari bit yang lebih tinggi. Digunakan untuk operasi aritmetika multiword

4) Equal

   Diset apabila hasil logika perbandingan sama

5) Overflow

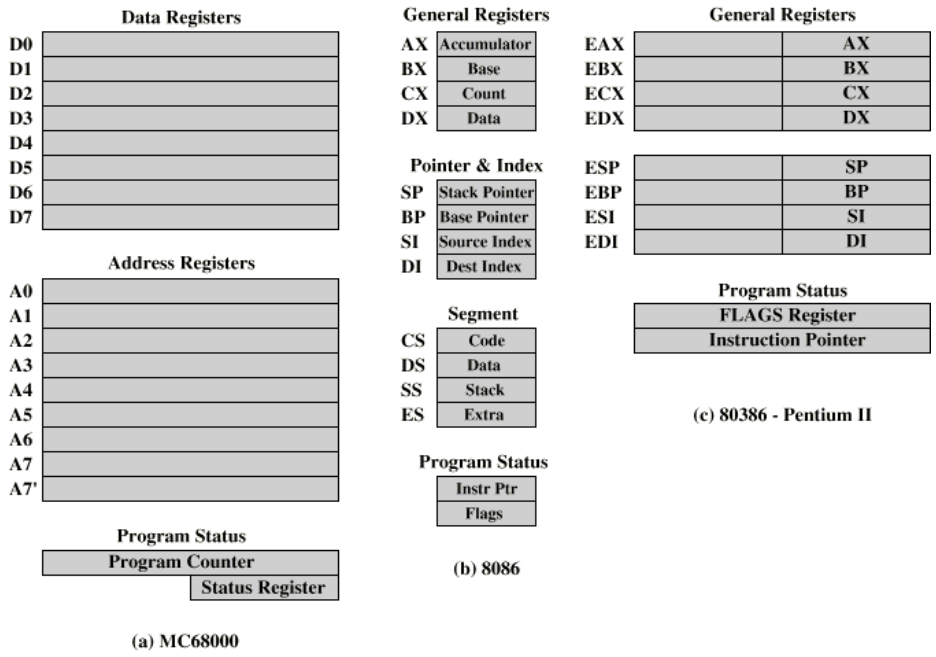
   Identifikasikan overflow aritmatika

6) Interrupt Enable/Disable

Status ijin terhadap interupt

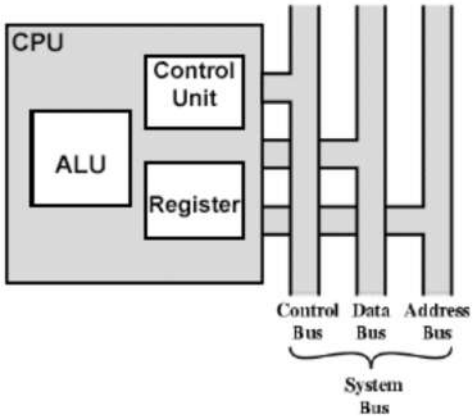
7) Supervisor

Mode previleged (ex. mode supervisor/user)

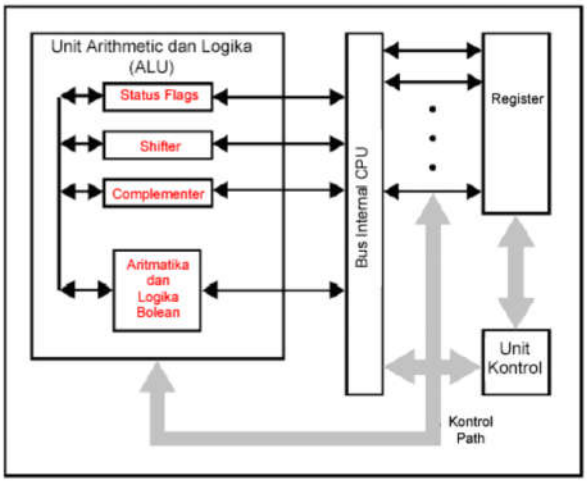


Gambar 1. Example Register Organizations

4. *CPU Interconnections,* adalah sistem koneksi dan bus yang menghubungkan komponen internal CPU, yaitu ALU, unit kontrol dan register – register dan juga dengan bus – bus eksternal CPU yang menghubungkan dengan sistem lainnya, seperti memori utama, piranti masukan/keluaran.



Gambar 2. Komponen Internal CPU

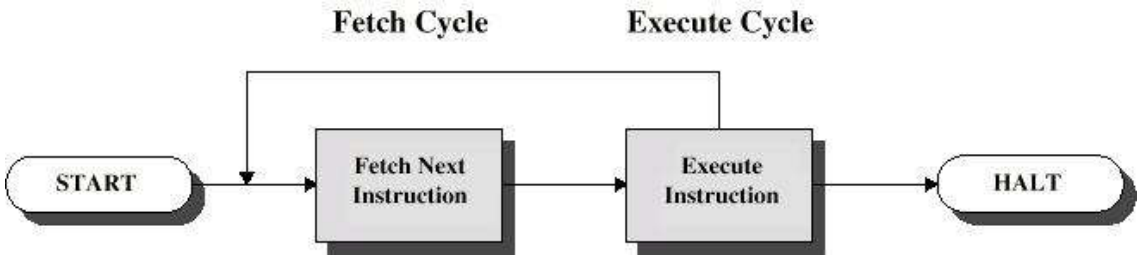


Gambar 3. Struktur detail internal CPU

CPU Function

Fungsi CPU yang pertama adalah menjalankan program-program yang disimpan dalam memori utama dengan cara mengambil instruksi-instruksi, menguji instruksi tersebut dan mengeksekusinya satu persatu sesuai alur perintah. Fungsi yang ke-dua adalah Pandangan paling sederhana proses eksekusi program adalah dengan mengambil pengolahan instruksi yang terdiri dari tiga langkah, yaitu :

1. Fetch: operasi pembacaan instruksi dan siklus pengambilan data ke memori atau register.
2. Execute: mengintrepetasikan opcode dan melakukan operasi yang diindikasikan atau operasi pelaksanaan instruksi
3. Interrupt : ketika interrupt diaktifkan atau telah terjadi simpan status proses yang sedang berjalan dan layani interupsi.



Gambar 4. Siklus instruksi

Siklus-siklus Fetch dan Eksekusi

Pada awal setiap siklus instruksi, CPU membaca instruksi dari memori. Pada CPU yang umum, suatu register yang disebut program counter (PC) dipakai untuk mengawasi instruksi yang akan dibca selanjutnya.

Intruksi yang dibaca akan dimuatkan ke dalam sebuah register di dalam CPU yang dikenal sebagai instruction register (IR). CPU menginterpretasikan intruksi dam melakukan aksi yang diperlukan. Secara umum, aksi-aksi ini dapat dibagi menjadi empat kategori:

- CPU-Memori: data dapat dipindahkan dari CPU ke memori atau dari memori ke CPU.
- CPU-I/O: Data dapat kea tau dunia luar denga pemindahan antara CPU dan modul I/O.
- Pengolahan data: CPU dapat membentuk sejumlah operasi aritmetik atau logic terhadap data.
- Control: Sebuah instruksi dapat menguabah urutan eksekusi (misalnya, insruksi lompat IAS, Tabel 2.1). Misalnya, CPU dapat membaca instruksi dari lokasi 149, yang menentukan bahwa instruksi berikutnya dibaca dari lokasi 182. CPU akan mengingat hal ini dengan menyetel program counter ke 182. Jadi, pada siklus fetch berikutnya, instruksi akan dibaca dari lokasi 182, bukannya 150.

CPU terdiri dari akumulator (AC) untuk menyimpan data secara sementara. Baik data dan instruksi panjangnya 16 bit. Format instruksi, menandakan bahwa akan terdapat sejumlah 24 = 16 op code yang berlainan dan sejumlah 212 = 4096 (4K) word memori yang dapat diamati secara langsung.

Diperlukan tiga buah instruksi, yang dapat dijelaskan sebagai tiga siklus fetch dan tiga eksekusi :

1. Program counter (PC) berisi 300 alamat instruksi pertama. Alamat ini dimuatkan ke dalam instruction register (IR). Perlu dicatat bahwa proses ini akan melibatkan penggunaan memory address register (MAR) dan memory buffer register (MBR). Untuk mudahnya, register-register intermediate-nya di abaikan.
2. 4 bit pertama di dalam IR mengindikasikan bahwa akumulator (AC) akan dimuatkan. 12 bit sisanya menentukan alamat, yaitu 940.
3. PC dinaikkan nilainya, dan instruksi berikutnya akan diambil.
4. Isi AC yang lama dan isi lokasi 941 ditambahkan, dan hasilnya disimpan di dalam AC.

5. PC dinaikkan nilainya, dan instruksi berikutnya akan diambil.
6. Isi PC akan disimpan pada lokasi 941.

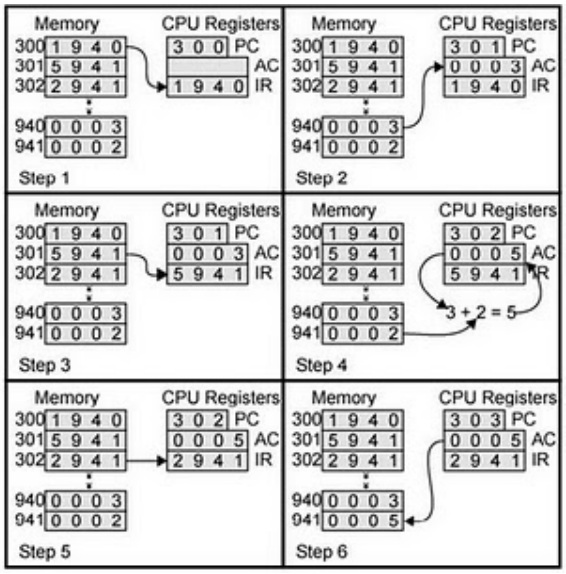
Instruksi PDP-11 yang diungkapkan secara simbolik sebagai ADD B, A menyimpan jumlah isi lokasi memori B dan A ke dalam lokasi memori A. Terjadi suatu siklus instruksi tunggal dengan langkah-langkah sebagai berikut :

1. Mengambil (fetch) instruksi ADD,
2. Membaca isi lokasi memori A ke dalam CPU.
3. Membaca isi lokasi memori B ke dalam CPU. Agar isi A tidak hilang, CPI harus memiliki sedikitnya dua buah register untuk menyimpan nilai-nilai memoti.
4. Menambahkan kegua nilai itu.
5. Menuliskan hasilnya dari CPU Kke lokasi memori A.

Jadi, siklus eksekusi untuk instruksi tertentu dapat melibatkan lebih dari sebuah referensi ke memori, juga, disamping referensi memori, suatu instruksi dapat menentukan suatu operasi I/O. Untuk sembarang siklus instruksi yang diketahui, sebagian keadaan dapat null dan lainnya dapat lebih dari sekali. Keadaan teersebut adalah :

- Instruction Address Calculation (aic): Menenetukan alamat instruksi berikutnya yang akan dieksekusi. Biasanya, hal ini melibatkan penambahan bilangan tetap ke alamat instruksi sebelumnya.
- Instruction Fetch (if): Membaca instruksi dari lokasi memorinya ke dalam CPU.
- Instruction Operation Decoding (oac): Bila operasi melibatkan referensi ke operand didalam memori atau dapat diperoleh melalui I/O, maka tentukan alamat operand.
- Operand Fetch (of): Ambil operand dari memori dan baca operand itu dari I/O.
- Data Operation (do): Bentuk operasi yang ditunjukkan di dalam instruksi.
- Operand Store (os): Tuliskan hasilnya ke dalam memori atau keluarkan ke I/O.

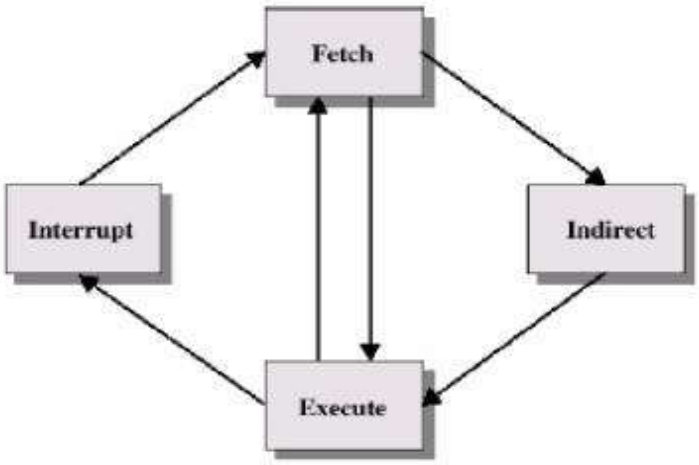
Untuk sebagian mesin, instruksi tunggal dapat menentukan operasi yang akan di bentuk pada suatu vector (array dimensi satu) bilangan-bilangan atau suatu string (array dimensi satu) karakter-karakter.



Gambar 4. Contoh eksekusi program

Siklus Tak Langsung

- Eksekusi sebuah instruksi dapat melibatkan sebuah operand atau lebih di dalam memori, yang masing-masing memori memerlukan akses memori
- Pengambilan alamat-alamat yang tak langsung dapat dianggap sebagai sebuah subsiklus instruksi

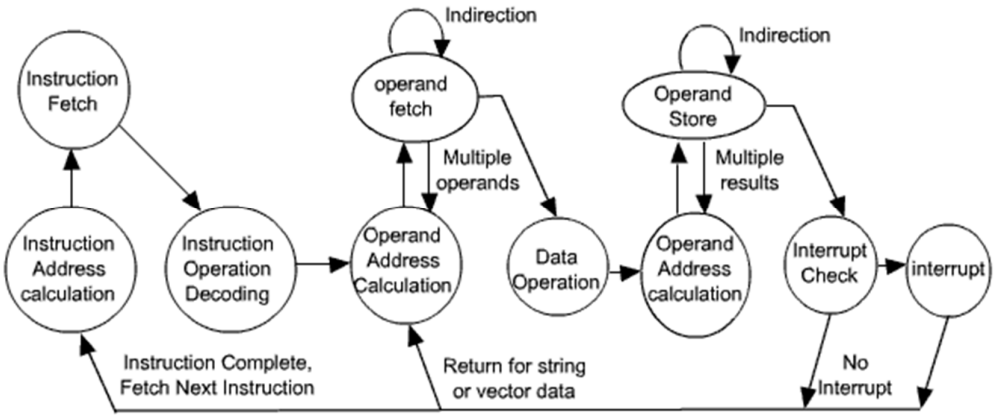


Gambar 5. Instruksi tidak langsung

Sifat siklus instruksi:

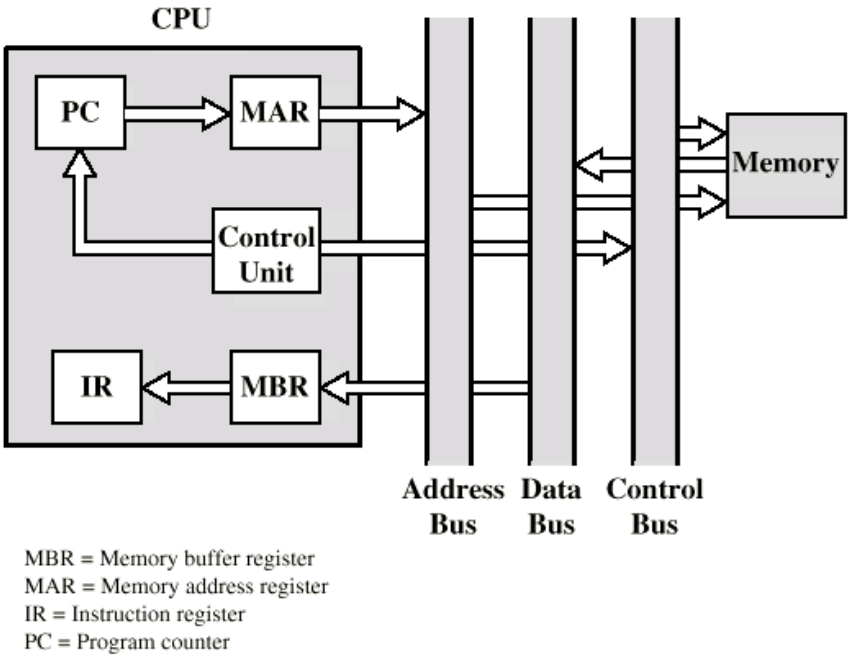
- Sekali instruksi terambil makan operand specifiernya harus diidentifikasi.

- Seluruh operand input di memori akan diambil sementara operand berbasis register tidak perlu diambil.
- Setelah eksekusi terjadi proses yang sama diperlukan untuk menyimpan hasilnya didalam memori.



Gambar 6. Instruction Cycle State Diagram

Aliran Data Siklus Fetch



Gambar 7. Data flow (fatch diagram)

- Urutan kejadian selama siklus instruksi tergantung pada rancangan CPU
- Umumnya : sebuah komputer yang menggunakan register memori alamat (MAR), register memori buffer (MBR),pencacah program (PC) dan register instruksi (IR).

Instruction Fetch

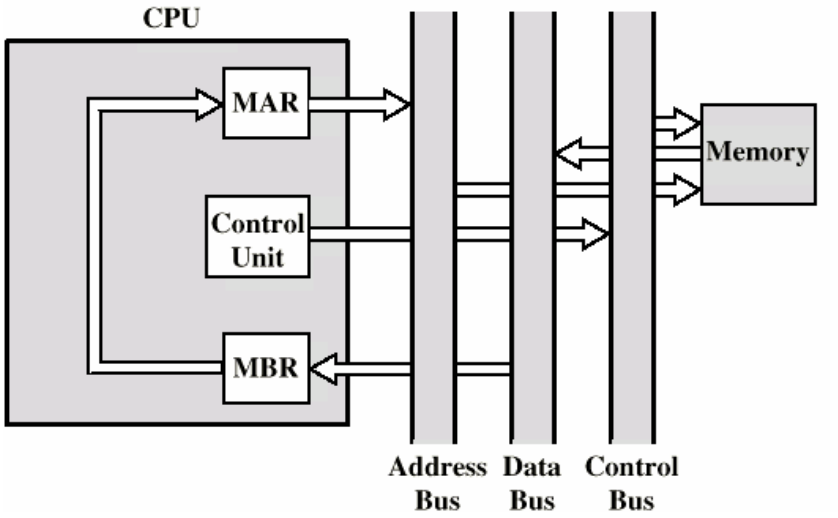
Prosesnya:

- Pada saat siklus pengambilan (fetch cycle), instruksi dibaca dari memori
- PC (Program Counter) berisi alamat instruksi berikutnya yang akan diambil
- Alamat ini akan dipindahkan ke MAR dan ditaruh di bus alamat
- Unit kontrol meminta pembacaan memori dan hasilnya disimpan di bus data dan disalin ke MBR dan kemudian dipindahkan ke IR
- PC naik nilainya 1, sebagai persiapan untuk pengambilan selanjutnya.

Selanjutnya:

- Siklus selesai, unit kontrol memeriksa isi IR
- Untuk menentukan apakah IR berisi operand specifier yang menggunakan pengalamatan tak langsung
- Apabila pengalamatan tak langsung, maka siklus tak langsung akan dijalankan.

Aliran Data Siklus Tak Langsung



Gambar 8. Data flow (indirect diagram)

- N bit paling kanan pada MBR, yang berisi referensi alamat, dipindahkan ke MAR
- Unit kontrol meminta pembacaan memori, agar mendapatkan alamat operand yang diinginkan ke dalam MBR.

Aliran Data Siklus Eksekusi

- Siklus pengambilan dan siklus tak langsung cukup sederhana dan dapat diramalkan

- Siklus instruksi (instruction cycle) mengambil banyak bentuk karena bentuk bergantung pada bermacam-macam instruksi mesin yang terdapat di dalam IR
- Siklus meliputi pemindahan data di antara register-register, pembacaan atau penulisan dari memori, I/O, dan atau penggunaan ALU

Aliran Data Siklus Eksekusi

- Siklus pengambilan dan siklus tak langsung cukup sederhana dan dapat diramalkan
- Siklus instruksi (instruction cycle) mengambil banyak bentuk karena bentuk bergantung pada bermacam-macam instruksi mesin yang terdapat di dalam IR
- Siklus meliputi pemindahan data di antara register-register, pembacaan atau penulisan dari memori, I/O, dan atau penggunaan ALU

Siklus Eksekusi

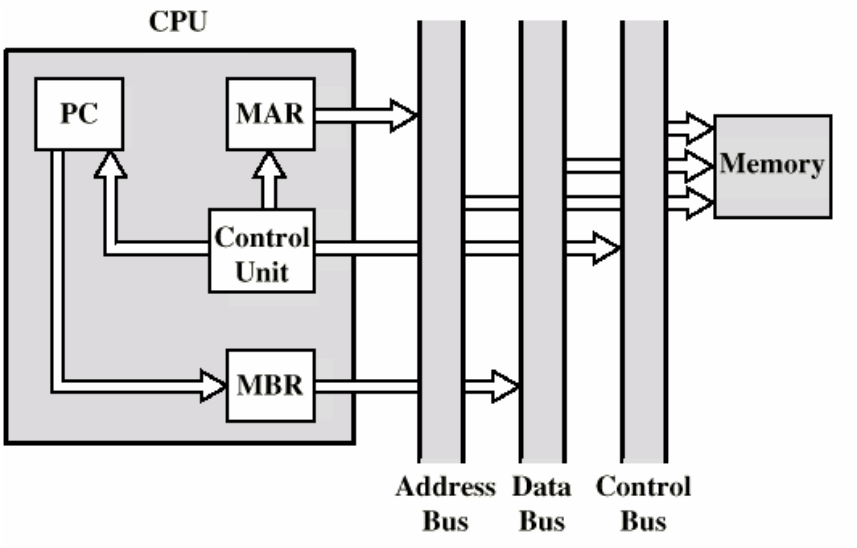
- ***Instruction Addess Calculation (IAC),*** yaitu mengkalkulasi atau menentukan alamat instruksi berikutnya yang akan dieksekusi. Biasanya melibatkan penambahan bilangan tetap ke alamat instruksi sebelumnya. Misalnya, bila panjang setiap instruksi 16 bit padahal memori memiliki panjang 8 bit, maka tambahkan 2 ke alamat sebelumnya
- ***Instruction Fetch (IF),*** yaitu membaca atau pengambil instruksi dari lokasi memorinya ke CPU
- ***Instruction Operation Decoding (IOD)****,* yaitu menganalisa instruksi untuk menentukan jenis operasi yang akan dibentuk dan operand yang akan digunakan
- ***Operand Address Calculation (OAC)****,* yaitu menentukan alamat operand, hal ini dilakukan apabila melibatkan referensi operand pada memori
- ***Operand Fetch (OF)****,* adalah mengambil operand dari memori atau dari modul I/O
- ***Data Operation (DO)****,* yaitu membentuk operasi yang diperintahkan dalam instruksi
- **Operand store (OS)**, yaitu menyimpan hasil eksekusi ke dalam memori

Aksi CPU

- *CPU – Memori*, perpindahan data dari CPU ke memori dan sebaliknya

- *CPU –I/O*, perpindahan data dari CPU ke modul I/O dan sebaliknya
- *Pengolahan Data*, CPU membentuk sejumlah operasi aritmatika dan logika terhadap data
- *Kontrol*, merupakan instruksi untuk pengontrolan fungsi atau kerja. Misalnya instruksi pengubahan urutan eksekusi.

Aliran Data Siklus Interupsi



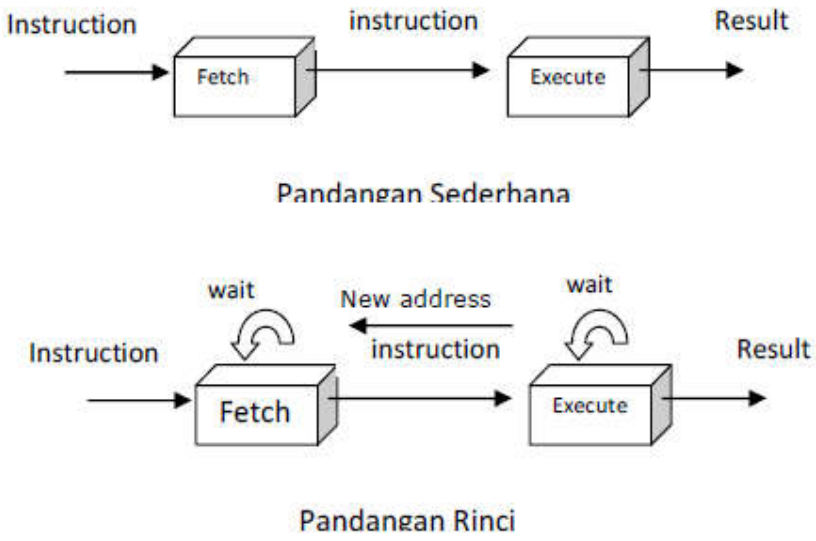
Gambar 9. Data flow (interrupt diagram)

- Isi PC saat itu harus disimpan sehingga CPU dapat melanjutkan aktivitas normal setelah terjadinya interrupt
- Cara : Isi PC dipindahkan ke MBR untuk kemudian dituliskan ke dalam memori
- Lokasi memori khusus yang dicadangkan untuk keperluan ini dimuatkan ke MAR dari unit kontrol
- Lokasi ini berupa stack pointer
- PC dimuatkan dengan alamat rutin interrupt
- Akibatnya, siklus instruksi berikutnya akan mulai mengambil instruksi yang sesuai

Pipeline

Mesin yang melaksanakan beberapa komputasi yang berbeda secar bersama-sama, namun pada saat itu setiap komputasi akan berada dalam tahapan eksekusi

yang berbeda.Input baru akan diterima pada sebuah sisi sebelum input yang diterima sebelumnya keluar sebagai output di sisi lainnya.



Gambar 10.instruksi pipeline

Tahapan Pipeline

- Tahapan pertama mengambil instruksi dan membuffernya.
- Ketika tahapan kedua bebas, tahapan pertama mengirimkan instruksi yang di bufferkan tersebut.
- Pada saat tahapan kedua sedang mengeksekusi instruksi, tahapan pertama memanfaatkan siklus memori yang tidak dipakai untukmengambil dan membufferkan instruksi berikutnya.

Efek Pipeline

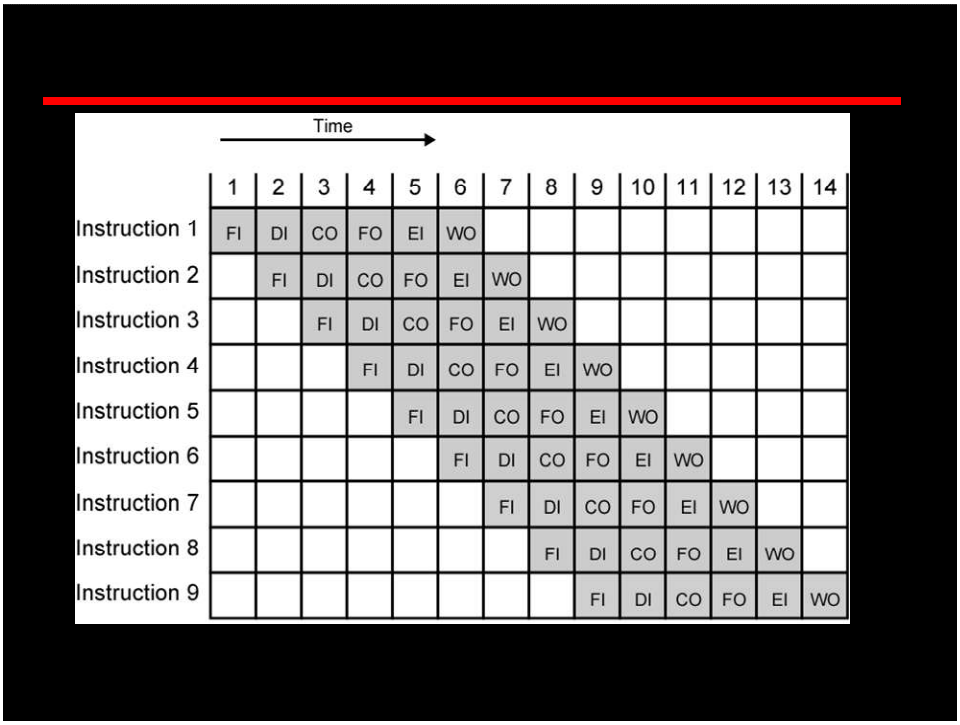
- Mempercepat eksekusi instruksi.
- Apabila tahapan pengambilan dan eksekusi memerlukan waktu yang sama, maka siklus instruksi akan berkurang menjadi separuhnya.

Two Stage Instruction Pipeline

2 tahapinstruksi pipeline

- Tahanpertamamengambilinstuksidanmem-buffer-kannya

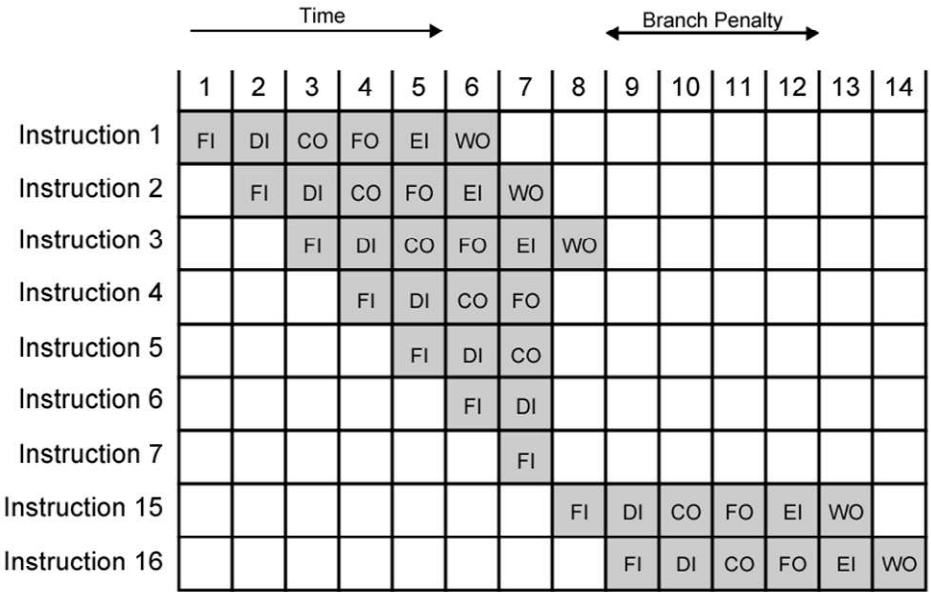
- Ketikatahapankeduabebas, tahapanpertamamengirimkaninstruksi yang dibufferkantersebut
- Pada saat tahapan kedua sedang mengeksekusi instuksi,tahapan pertama memanfaatkan siklus memori yang tidak bisa dipakai untuk mengambil dan mem-buffer-kan instruksi berikutnya
- Proses ini disebut instruksi prefetch atau fetch overlap



Faktormenghambat peningkatan kinerja

- Keenam tahapan memiliki durasi yang tidak sama, terjadi waktu tunggu pada beberapa tahapan pipeline
- Instruksi percabangan bersyarat, yang dapat menggagalkan beberapa pengambilan instruksi

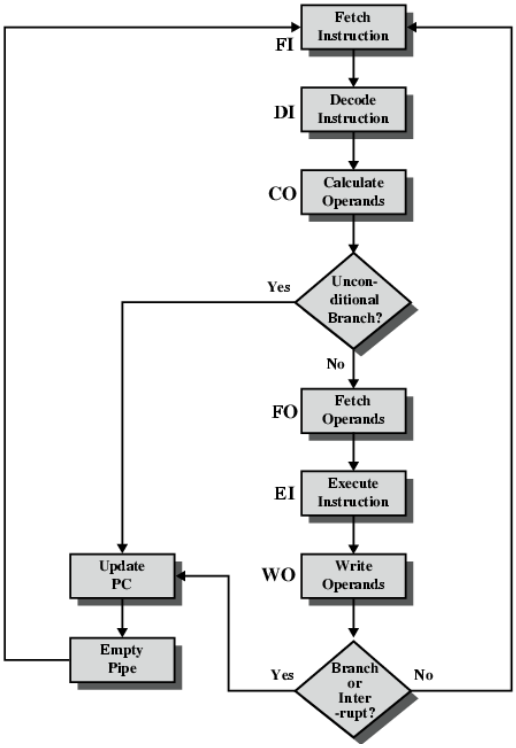
The Effect of a Conditional Branch on Instruction Pipeline Operation



Asumsi :

- Instruksi 3 adalah percabangan bersyarat instruksi 15
- Sampai saat instruksi di eksekusi, tidak terdapat cara untuk mengetahui instruksi mana yang akan terjadi kemudian
- Instruksi 4 sampai 14 tidak dilakukan eksekusi sehingga data harus dibersihkan dari jalurnya
- Eksekusi dilanjutkan saat percabangan ke instruksi 15 sudah sampai

Six Stage Instruction Pipeline



Fetch Instruction(FI)

- Membacainstruksiberikutnyakedalam buffer

Decode Instruction (DI)

- Menentukanopcodedan operand specifier

Calculate Operand(CO)

- Menghitungalamatefektifseluruh operand sumber. Hal inimungkinmelibatkan displacement, register indirect, ataubentukkalkulasialamatlainnya

Fetch Operand(FO)

- Mengambilsemua operand darimemori. Operand-operand yang berada di register tidakperludiambil

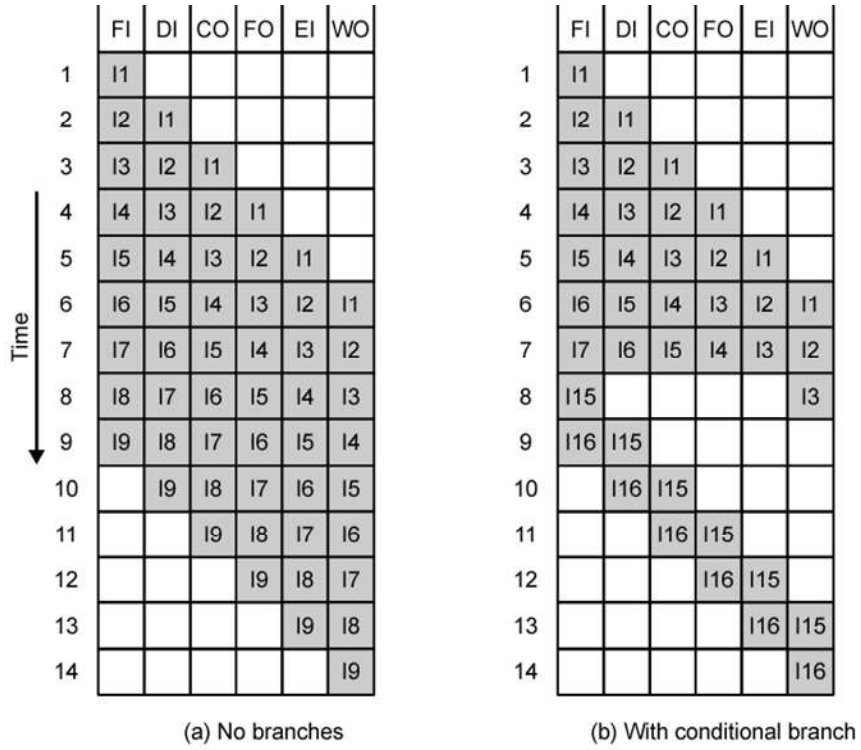
Execute Instruction(EI)

- Melakukanoperasi yang diindikasikanmenyimpanhasilnya

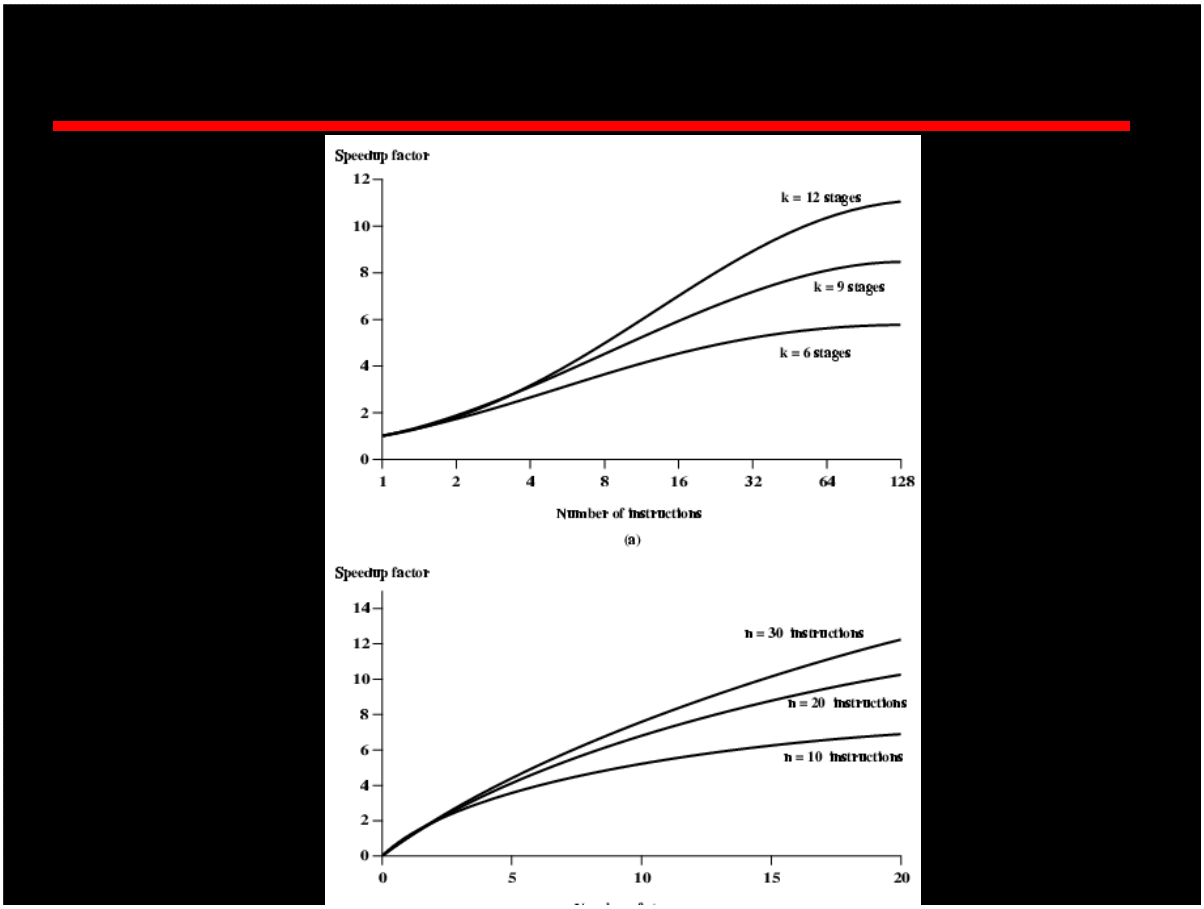
Write Operand(WO)

- Menyimpanhasilnya di dalam memory

Alternative Pipeline Depiction



- Menjaminterjadinyaaliraninstruksi yang stabil
- Kestabilanakanterganggusaatinstruksimengalamipercabangankarenabelum bisa menentukantujuanpercabangantersebut
- Beberapametodependekatanmasalahdigunakanuntukmengatasihaltersebut, antara lain multiple streams, prefetch branch target, loop buffer, branch prediction, delayed branch



Grafik untuk Speedup factors pada instruksi pipline

a. Speedup factor terhadap setiap instruksi yang dipengaruhi oleh jumlah stage(tahap)
b. Speedup factor terhadap setiap stage(tahap) yang dipengaruhi oleh jumlah instruksi

- Multiple Streams
  - Keduainstruksipercabangandiambildengan 2 buah stream
  - Kelemahan :
  - Adanyapersaingandalammengakses register danmemoriuntukdimasukkandalam pipeline
  - Bilaada
- Prefetch Branch Target

  Apabilapencabanganbersyarattelahdiketahui

  Prosesnya :

  Dilakukanpengambilanawal (prefetch) terhadapinstruksisetelahpencabangandan target pencabangan. Diterapkanpada IBM 360/91.

  Masalah :

  Diperlukan buffer dan registeruntukprefetch

- Loop buffer
  - Apabilaterdapatpencabanganmakaperangkatkerasmemeriksaapakah target pencabangantelahadadalam buffer, bilatelahadamakainstruksiberikutnyadiambildari buffer.
  - Perbedaandenganprefetchadalahpada loop buffer akanmembufferinstruksikedepandalamjumlah yang banyak, sehinggabila target tidakberjauhanlokasinyamakasecaraotomatistelahterbuffer.
  - Terkesanteknikiniseperti cache memori, namunterdapatperbedaankarena loop buffer masihmempertahankanurutaninstruksi yang diambilnya

- Branch prediction
  - Instruksi computer seringkali terjadi berulang.Sehingga :
  - Teknikprediksiinijugaditerapkandalampengambilaninstruksipada cache memori.
  - Diperlukanalgoritmakhususuntukmelakukanprediksitersebut.
  - Patokanmemprediksi target pencabangan
    - Penganalisaaneksekusi – eksekusi yang telahterjadidanaspeklokalitas.
    - Aspeklokalitasmemoriadalahkecenderunganpenyimpananinstruksi yang berhubungandalamtempat yang berdekatan
- Delayed branching
  - Eksekusipadatahapan yang melibatkanpencabanganakandilakukanpenundaan proses beberapasaatsampaididapatkanhasilpencabangan.
  - Namuntahapan pipelining lainnyadapatberjalanseiringpenundaantersebut.
  - Teknikpenundaaninimenggunakaninstruksi NOOP

Multiple Streams

- Memiliki 2 pipeline
- Prefetch setiapbranch(cabang) kesebuah pipeline yang terpisah
- Gunakan pipeline yang tepat
- Leads to bus & register contention
- Beberapa branch(cabang) mengakibatkan pipaline yang diperlukan lebihbanyak

Prefetch Branch Target

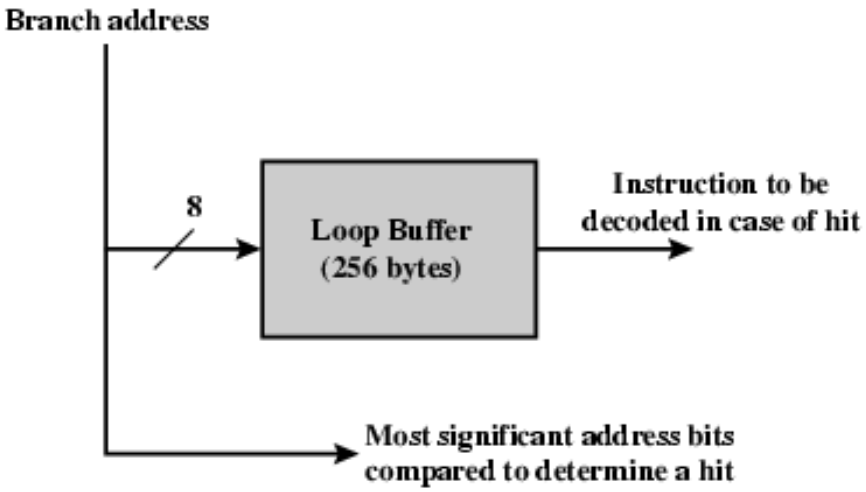
- Apabilapencabanganbersyarattelahdiketahui
  Prosesnya :Dilakukanpengambilanawal (prefetch) terhadapinstruksisetelahpencabangandan target pencabangan.
- Simpan target sampai branch dieksekusi
- Diterapkanpada IBM 360/91.

-

Loop Buffer

- Memori yang sangatcepat
- Maintained dilakukansaattahap fetch pada pipeline
- Check buffer sebelummelakukan fetchdari memory
- Sangatbagusuntukloopsatau jumps yang sederhana
- c.f. cache
- Diterapkanoleh CRAY-1

Loop Buffer Diagram



Loop Buffer Diagram

Apabila terdapat percabangan maka perangkat keras melakukan pemeriksaan terlebih dahulu pada buffer. Apabila target percabangan telah ada dalam buffer, maka instruksi berikutnya diambil dari buffer. Tidak perlu dari memory lagi. *Loop Buffer* akan melakukan buffer instruksi ke depan dalam jumlah yang banyak, sehingga bila target tidak berjauhan lokasinya maka secara otomatis telah terbuffer.*Loop Buffer* masih mempertahankan urutan instruksi yang diambilnya

Branch Prediction (1)

- Prediksitidakpernahdiambil
  - Asumsibahwa jump tidakakanterjadi
  - selalu fetchuntukinstruksiselanjutnya
  - 68020 & VAX 11/780
  - VAX tidakakanmelakukanprefetchsetelah branch(percabangan)sebuahhalamanmenghasilkankesalahan (O/S v CPU design)
- Prediksiselaludiambil
  - Asumsibahwa jump akanterjadi
  - Selalumelakukaninstruksi fetch target

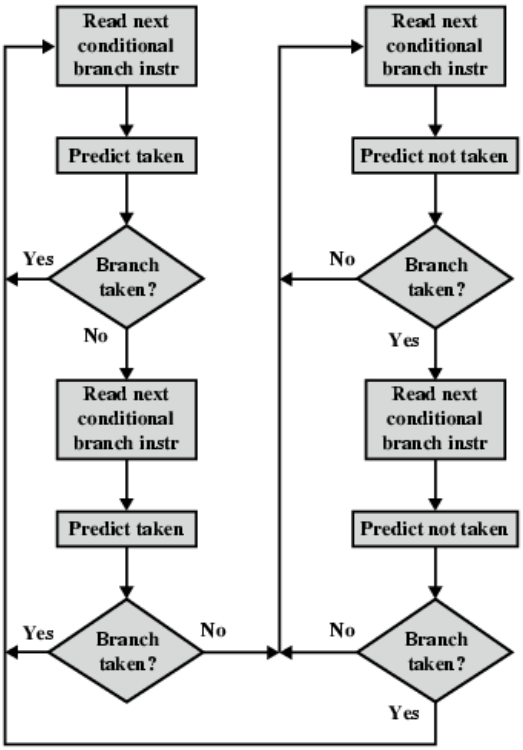
Branch Prediction (2)

- PrediksiolehOpcode
  - Beberapa instruksi lebih banyak menghasilkan sebuah jump dibandingkan thers
  - Dapat mencapai 75% berhasil
- Taken/Not taken switch
  - Berdasarkan pada history sebelumnya
  - Bagus untuk loops

Branch Prediction (3)

- Penundaan Branch
  - Melakukan jump jika harus dilakukan
  - Menyusun kembali instruksi-instruksi

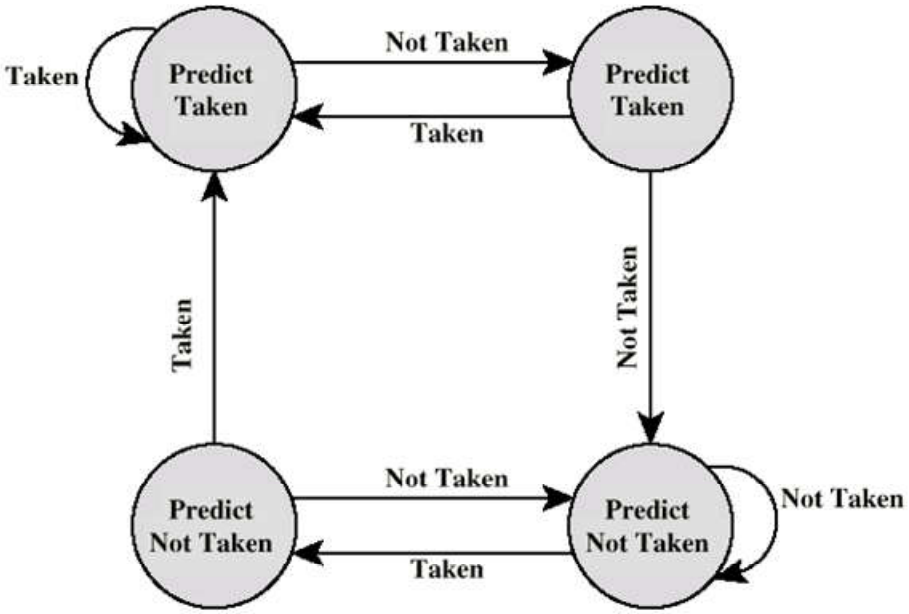
Branch Prediction Flowchart



Branch Prediction Flowchart

- Membaca instruksi kondisi percabangan
- Menerima prediksi
- Jika berhasil menerima prediksi akan kembali membaca instruksi kondisi percabangan semula, jika tidak akan membaca instruksi kondisi percabangan selanjutnya
- Untuk kondisi yang berbeda yaitu tidak menerima prediksi, jika benar akan membaca instruksi kondisi percabangan selanjutnya namun jika salah akan kembali membaca instruksi kondisi percabangan sebelumnya

Branch Prediction State Diagram



Branch Prediction State Diagram

Membuat prediksi yang baik untuk instruksi yang akan dieksekusi selanjutnya dan mulai yangsatu ke bawah pipeline.

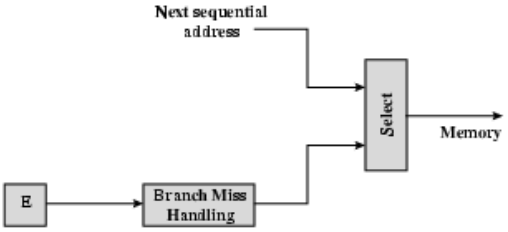
- Prediksi Statis: membuat prediksi tanpa mempertimbangkan runtime history dari program

  Cabang tidak pernah diambil

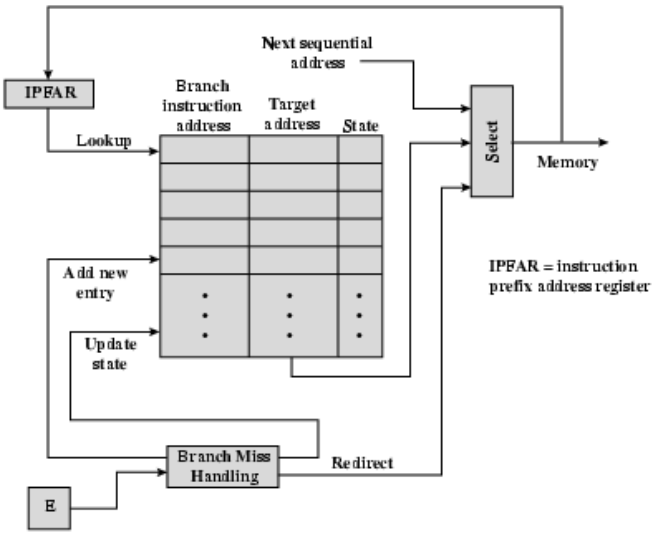
  Cabang selalu diambil

  Prediksi berdasarkan opcode
- Prediksi Dinamis : melacak sejarah cabang kondisional dalam program.

Dealing With Branches



(a) Predict never taken strategy



(b) Branch history table strategy

Dealing With Branches

Dalam sebuah siklus instruksi yang terjadi pada CPU, umumnya waktu eksekusi akan lebih lama dibandingkan dengan pengambilan instruksi. Karena eksekusi ini akan meliputi pembacaan dan penyimpanan operand serta kinerja sejumlah operasi sehingga tahapan pengambilan instruksi mungkin perlu menunggu beberapa saat sebelum mengosongkan buffer-nya.

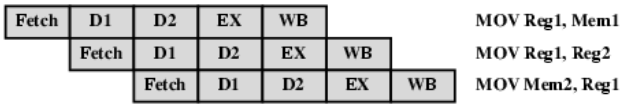
Dengan adanya instruksi percabangan bersyarat maka akan membuat alamat instruksi berikutnya yang akan diambil tidak diketahui. Tahapan pengambilan harus menunggu sampai menerima alamat instruksi berikutnya dari tahapan eksekusi. Dengan demikian tahap eksekusi harus menunggu pada saat fetch. Kestabilan akan tergangggu saat instruksi mengalami percabangan karena belum bisa ditentukan tujuan percabangan tersebut.

- Fetch
  - Dari cache ataueksternal memory
  - Meletakkansatudariduaprefetch buffers 16-byte
  - Mengisi buffer dengan data barusegerasetelah data lama digunakan
  - Rata-rata 5 instruksi fetched setiapmengisi

- Bebasdaritahap yang lainuntukmenyimpan buffers denganpenuh

- Decode stage 1
  - InformasiOpcodedan address-mode
  - Pertama kali 3 bytes instruksi
  - Dapatlangsungketahap D2 untukmendapatistirahat
- Decode stage 2
  - Memperluasopcodekesinyalkontrol
  - Perhitungankompleks address modes
- Execute
  - OperasiALU, aksescache, memperbaharui register
- Writeback
  - Memperbaharui registers dan flags
  - Berakhirdenganmengirimke cache dan bus antarmuka write buffers

80486 Instruction Pipeline Examples



(a) No Data Load Delay in the Pipeline

(b) Pointer Load Delay

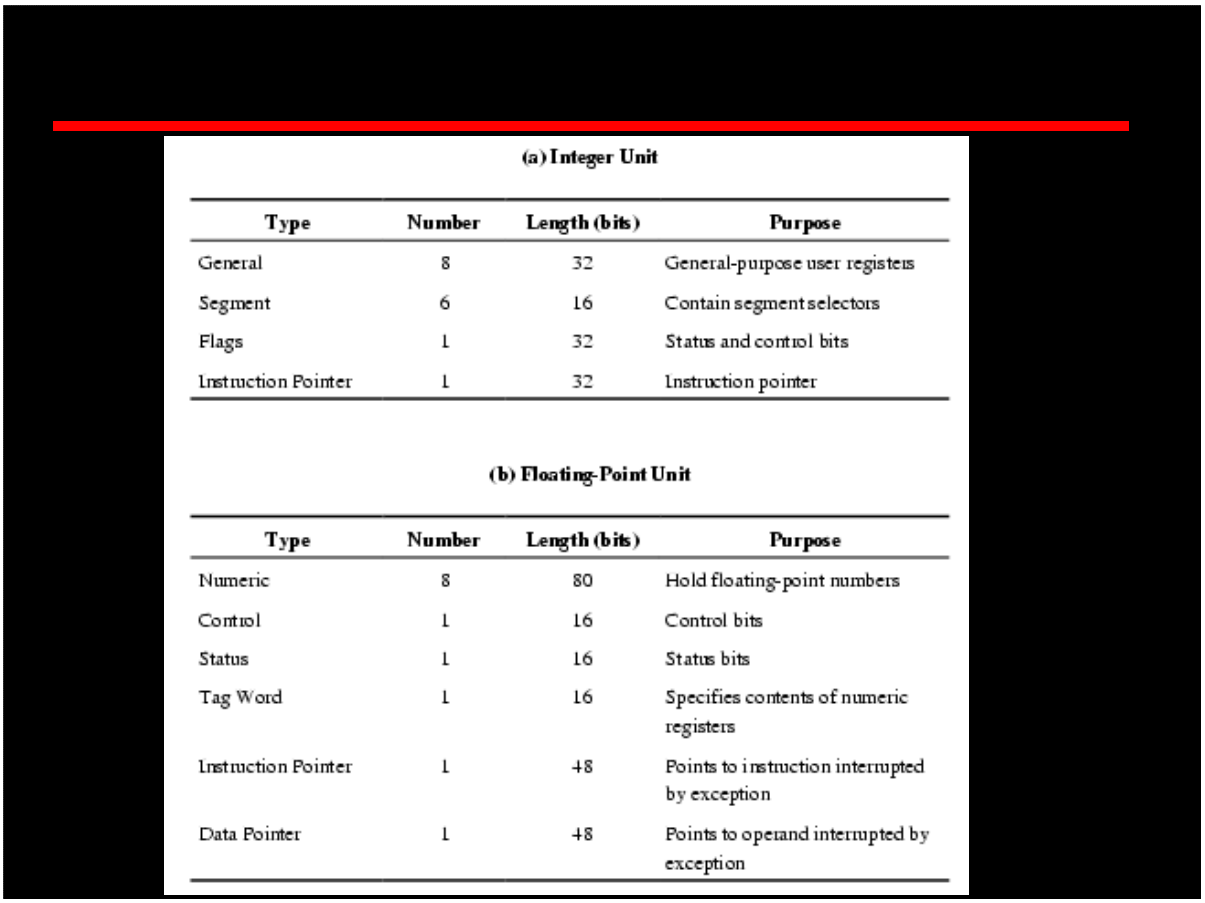
(c) Branch Instruction Timing

80486 Instruction Pipeline Examples

Instruction Pipeline adalah teknik yang digunakan dalam desain komputer dan perangkat elektronik digital untuk meningkatkan instruksi throughput mereka (jumlah instruksi yang dapat dieksekusi dalam unit waktu). Pipelining tidak mengurangi waktu yang dibutuhkan

untuk menyelesaikan instruksi, hal itu meningkatkan jumlah instruksi yang dapat diproses sekaligus, sehingga mengurangi penundaan antara selesai petunjuk

a. Tidak ada data yang mengalami penundaan saat dimuat
b. Penundaan saat memuat pointer
c. Pewaktuan instruksi branch(percabangan)



(a) Integer Unit

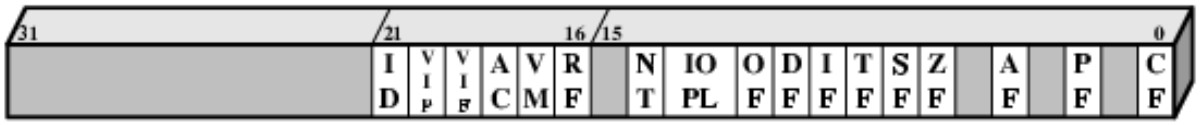
Type	Number	Length (bits)	Purpose
General	8	32	General-purpose user registers
Segment	6	16	Contain segment selectors
Flags	1	32	Status and control bits
Instruction Pointer	1	32	Instruction pointer

(b) Floating-Point Unit

Type	Number	Length (bits)	Purpose
Numeric	8	80	Hold floating-point numbers
Control	1	16	Control bits
Status	1	16	Status bits
Tag Word	1	16	Specifies contents of numeric registers
Instruction Pointer	1	48	Points to instruction interrupted by exception
Data Pointer	1	48	Points to operand interrupted by exception

Pada dasarnya, seluruh golongan/jenis Pentium 4 desktop termasuk prosesor 32 bit, menggunakan mikroarsitektur NetBurst, dan dilengkapi dukungan teknologi MMX, SSE, dan SSE2. Pada jenis-jenis tertentu diperlengkapi tambahan dukungan teknologi yang lain, misalnya hyper-threading, Intel 64 (Intel's x86-64 implementation), XD bit (an NX bit implementation), EIST (Enhanced Intel SpeedStep Technology), Virtualization Technology Teknology.

EFLAGS Register



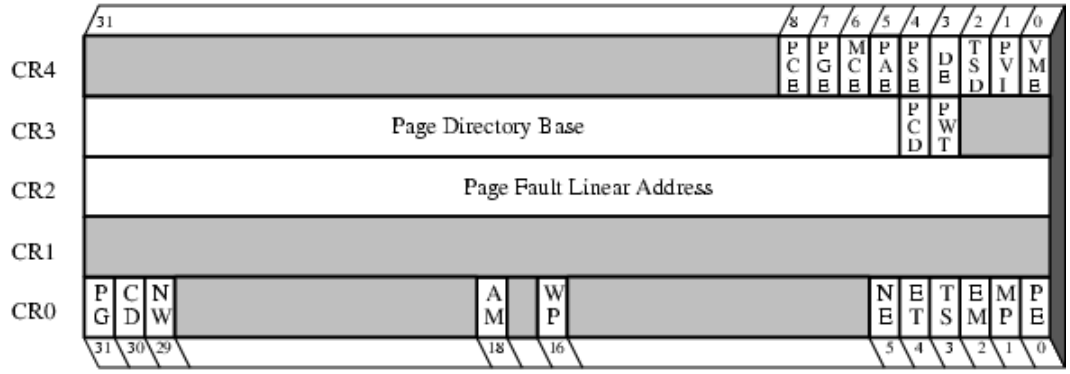
ID = Identification flag
VIP = Virtual interrupt pending
VIF = Virtual interrupt flag
AC = Alignment check
VM = Virtual 8086 mode
RF = Resume flag
NT = Nested task flag
IOPL = I/O privilege level
OF = Overflow flag
DF = Direction flag
IF = Interrupt enable flag
TF = Trap flag
SF = Sign flag
ZF = Zero flag
AF = Auxiliary carry flag
PF = Parity flag
CF = Carry flag

EFLAGS Register

EFLAGS Register adalah status register di Intel x86 mikroprosesor yang berisi keadaan saat ini prosesor. Register ini adalah 16 bit lebar. Penerus, yang EFLAGS dan register RFLAGS, yang 32 bit dan 64 bit lebar, masing-masing. Register yang lebih luas mempertahankan kompatibilitas dengan pendahulu mereka yang lebih kecil.

Pengujian, jika bit dalam Flags-register ada, yang hanya ada di prosesor tertentu adalah metode untuk menentukan apa Processor diinstal (Misalnya Alignment-Flag hadir pada 486 dan di atas, sehingga jika bisa toggled CPU adalah 486 atau lebih tinggi yang lain 386). Metode ini adalah salah satu standar untuk membedakan prosesor 186-586. Para 8086 dan 186 dapat dibedakan dengan bitshift CL- 32 (yang tidak berpengaruh pada 186 dan di atas menyebabkan hanya 5 bit signifikan lebih rendah). Para 586 dan Prosesor di atas dapat dibedakan dengan sebuah entri di CPUID kembali nilai-nilai.

Control Registers



PCE = Performance Counter Enable
PGE = Page Global Enable
MCE = Machine Check Enable
PAE = Physical Address Extension
PSE = Page Size Extensions
DE = Debug Extensions
TSD = Time Stamp Disable
PVI = Protected Mode Virtual Interrupt
VME = Virtual 8086 Mode Extensions
PCD = Page-level Cache Disable
PWT = Page-level Writes Transparent

PG = Paging
CD = Cache Disable
NW = Not Write Through
AM = Alignment Mask
WP = Write Protect
NE = Numeric Error
ET = Extension Type
TS = Task Switched
EM = Emulation
MP = Monitor Coprocessor
PE = Protection Enable

Register control adalah register prosesor yang mengubah atau mengontrol perilaku umum dari sebuah CPU atau perangkat digital lainnya. Tugas umum dilakukan oleh register kontrol termasuk mengganggu kontrol, switching mode pengalamatan , paging kontrol, dan Coprocessor kontrol.

CR0

Register CR0 adalah 32 bit panjang pada 386 dan lebih tinggi prosesor. Pada x86-64 prosesor dalam modus lama , itu (dan register kontrol lainnya) adalah 64 bit panjang. CR0 memiliki bendera berbagai kontrol yang memodifikasi operasi dasar dari prosesor

CR1

Dilindungi

CR2

Berisi nilai disebut Page Patahan Alamat Linear (PFLA). Ketika terjadi kesalahan halaman, alamat program mencoba untuk mengakses disimpan dalam register CR2.

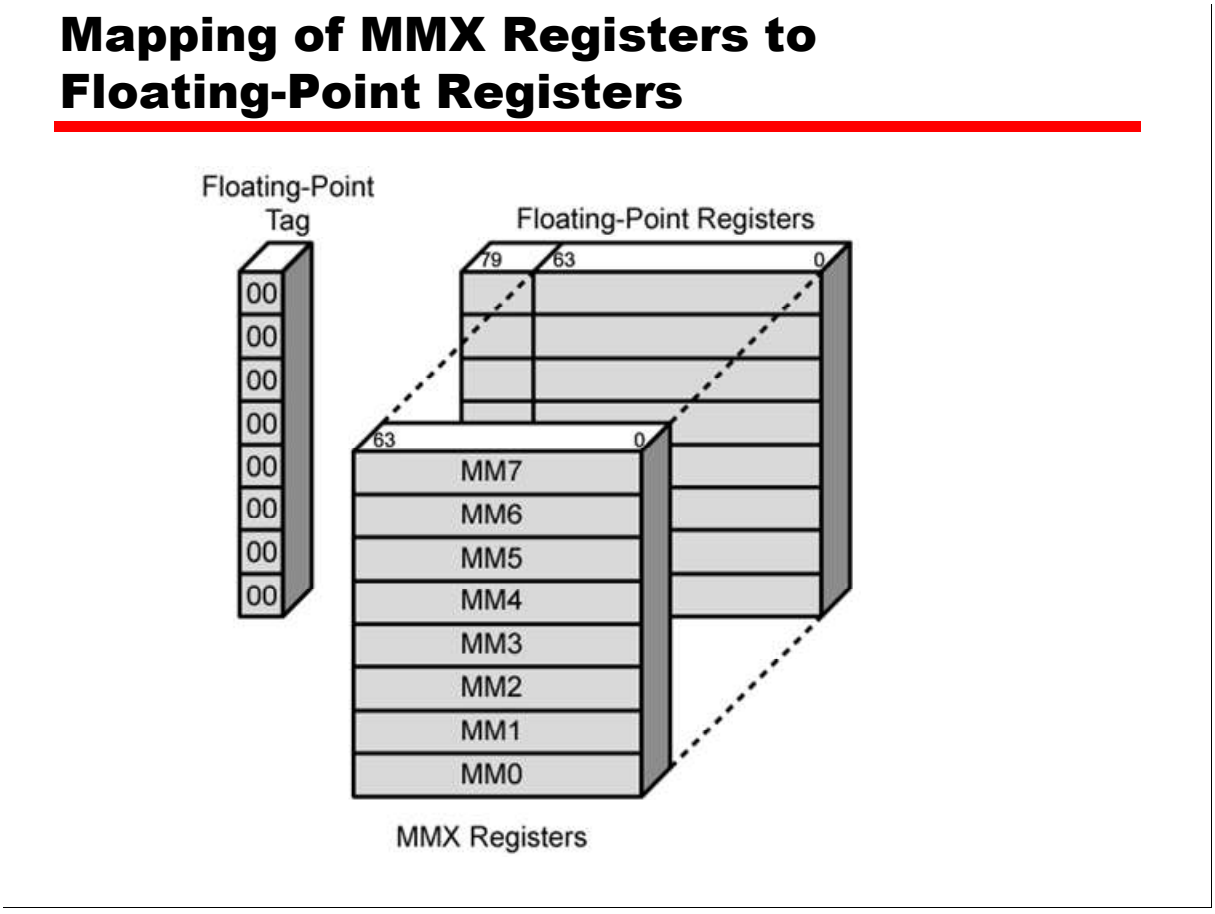
CR3

CR3 memungkinkan prosesor untuk menerjemahkan alamat virtual ke alamat fisik dengan menempatkan direktori halaman dan tabel halaman untuk tugas saat ini. Biasanya, 20 atas bit CR3 menjadi *direktori halaman base register* (PDBR), yang menyimpan alamat fisik dari entri direktori halaman pertama.

CR4

Digunakan dalam mode dilindungi untuk mengontrol operasi seperti virtual-8086 dukungan, memungkinkan I / O Breakpoints, ukuran dan ekstensi halaman pengecualian periksa mesin

- MMX menggunakanbeberapatipe date 64 bit
- Menggunakan 3 bit register address fields
  - 8 registers
- No MMX register tertentu
  - Menggunakandibawah 64 bit pada floating point registers



Nilai ditulis ke MMX register menggunakan instruksi MMX juga ditampilkan di salah satu dari delapan register floating-point (bit 63-0). Bidang eksponen yang sesuai dengan register floating-point (78-64 bit) dan bit tandanya (bit 79) ditetapkan untuk sesuatu (11s)

Mantissa dari nilai floating-point ditulis ke register floating-point dengan instruksi floating-pointjuga muncul dalam sebuah register MMX

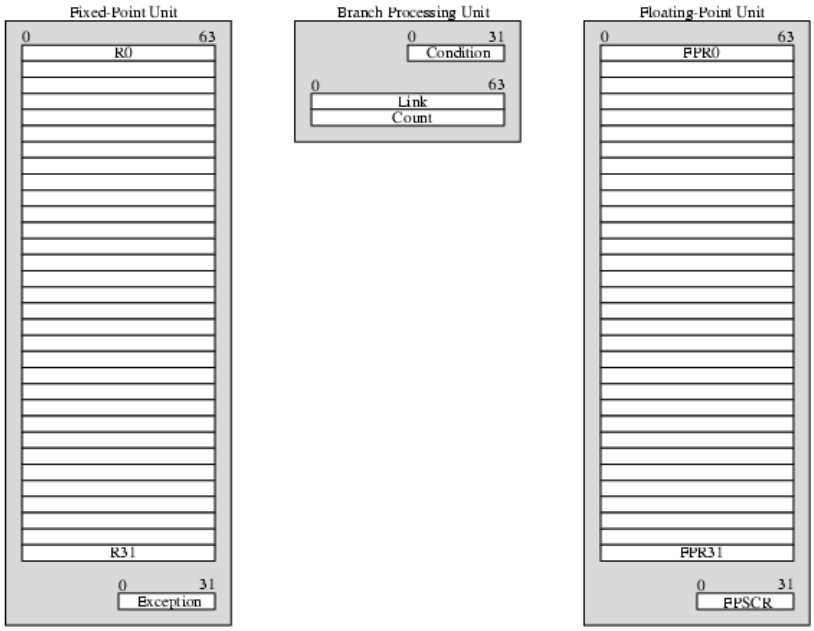
Peta register MMX ke lokasi fisik dari register floating-point. Pemetaan register MMX adalah tetap dan tidak berubah bila TOS (tempat Top Of Stack pada status floating-point, bit11-13) berubah

- Interrupts
  - Maskable

    Perangkat keras interrupt yang dapat diabaikan dengan menetapkan sedikit dalam sebuah topeng interupsi mendaftar 's (AKB) topeng bit
  - Nonmaskable

    perangkat keras interrupt yang tidak memiliki suatu yang terkait bit-topeng, sehingga tidak dapat diabaikan. NMIs sering digunakan untuk timer, terutama timer pengawas
- Pengecualian
  - Mendeteksi Processor
  - Pemrograman
- Interrupt vector table
  - Setiap jenis interrupt diberi sebuah nomor
  - Tabel vector memiliki indeks
  - 256 + 32 bit interrupt vectors
- 5 klas prioritas

PowerPC User Visible Registers



PowerPC User Visible Registers

Register yang dapatdireferensikandenganmenggunakanbahasamesin yangdieksekusi CPU. Register inimemungkinkanpemrogramanbahasamesindanbahasa assembler danmeminimalkanreferensi main memoridengancaramengoptimasipenggunaan register

_ Kategorinya :

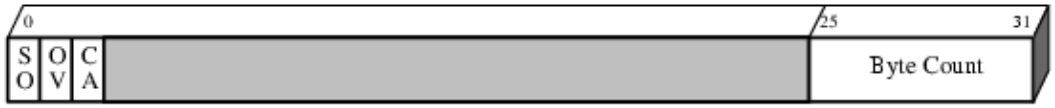
_ General Purpose

_ Data

_ Alamat

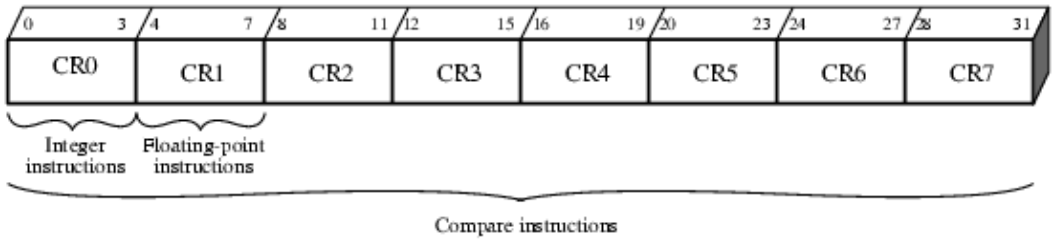
_ Kode-kodeKondisi

PowerPC Register Formats



SO = Summary overflow: set to 1 to indicate an overflow occurred during the exection of an instruction; remains 1 until reset by software
OV = Overflow: set to 1 to indicate an overflow occurred during the exection of an instruction; res to 0 by next instruction if there is no overflow
CA = Carry: set to 1 to indicate carry out of bit 0 during the execution of an instruction
Byte Count = Specifies number of bytes to be transferred by Load/Store String indexed instruction

(a) Fixed-Point Exception Register (XER)



(b) Condition Register

SO = Ringkasan overflow. setke 1 untukmengindikasikan overflow terjadiselamaexection dari instruksi; tetap 1 sampai direset oleh software(perangkatlunak)

OV = Overflow; set ke 1 untukmengindikasikan overflow terjadiselama exection dari instruksi; res ke 0 pada instruksi berikutnyajika tidakada overflow

CA = Carry; set ke 1 untukmenunjukkanmelaksanakan bit 0 selama exection dariinstruksi

Byte Count = menentukan jumlah byte yang akanditransferoleh instruksi string diindeks Load / Store

REFERENSI

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

PROPAGASI

A. Latihan dan Diskusi (Propagasi vertical dan Horizontal)

12.1 What general roles are performed by processor registers?

12.2 What categories of data are commonly supported by user-visible registers?

12.3 What is the function of condition codes?

12.4 What is a program status word?

12.5 Why is a two-stage instruction pipeline unlikely to cut the instruction cycle time in half, compared with the use of no pipeline?

B. Pertanyaan (Evaluasi mandiri)

12.1 a. If the last operation performed on a computer with an 8-bit word was an addition in which the two operands were 00000010 and 00000011, what would be the value of the following flags?

- Carry
- Zero
- Overflow
- Sign
- Even Parity
- Half-Carry

b. Repeat for the addition of (twos complement) and .

12.2 Repeat Problem 12.1 for the operation A B, where A contains 11110000 and B contains 0010100.

12.3 A microprocessor is clocked at a rate of 5 GHz.

a. How long is a clock cycle?

b. What is the duration of a particular type of machine instruction consisting of three clock cycles?

C. QUIZ -mutiple choice (Evaluasi)

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