**Tugas Organisasi dan Arsitektur Komputer**

**SET INSTRUKSI**

Fahmawati Nur M\_L200150048\_A

A. Pengertian Set Instruksi

* Set instruksi (instruction set) adalah sekumpulan lengkap instruksi yang dapat dimengerti oleh sebuah CPU.
* Instruksinya berbentuk *machine code* (bahasa mesin), aslinya seluruhnya bilangan biner.
* Untuk programmer, biasanya digunakan representsi yang lebih mudah dimengerti. Bahasa yang dimengerti manusia dikenal dengan bahasa Assembly.

B. Kumpulan Perintah/Instruksi yang Dapat Dimengerti Oleh Sebuah CPU

a.k.a. dengan sebuah kamus berisi daftar perintah apa saja yang dapat dilakukan (didukung) oleh sebuah prosesor, dan biasanya terikat dengan sebuah keluarga arsitektur prosesor tertentu (misal x86, x64).

Jenis – jenis Instruksi :

1. CISC (Complex Instruction-Set Computer)

CISC merupakan suatu bentuk arsitektur komputer yang mana pada setiap instruksi akan menjalankan beberapa operasi tingkat rendah, seperti pengambilan dari memory, operasi aritmetika, dan penyimpanan ke dalam memory, semuanya sekaligus hanya di dalam sebuah instruksi.

Karakteristik CISC :

* Sarat informasi memberikan keuntungan di mana ukuran program-program yang dihasilkan akan menjadi relatif lebih kecil, dan penggunaan memory akan semakin berkurang. Karena CISC inilah biaya pembuatan komputer pada saat itu (tahun 1960) menjadi jauh lebih hemat.
* Dimaksudkan untuk meminimumkan jumlah perintah yang diperlukan untuk mengerjakan pekerjaan yang diberikan. (Jumlah perintah sedikit tetapi rumit) Konsep CISC menjadikan mesin mudah untuk diprogram dalam bahasa rakitan.

Ciri-ciri :

* Jumlah instruksi banyak
* Banyak terdapat perintah bahasa mesin
* Instruksi lebih kompleks

Pengaplikasian CISC yaitu pada AMD dan Intel. Contoh-contoh prosesor CISC adalah :

* System/360VAX.
* PDP-11.
* Varian Motorola 68000.
* CPU AMD dan Intel x86.

2. **RISC *(Reduced Instruction Set Computer)***

*RISC singkatan dari Reduced Instruction Set Computer*. Merupakan bagian dari arsitektur mikroprosessor, berbentuk kecil dan berfungsi untuk negeset instruksi dalam komunikasi diantara arsitektur yang lainnya.

Karakteristik :

* One cycle execution time : satu putaran eksekusi. Prosessor RISC mempunyai CPI (clock per instruction) atau waktu per instruksi untuk setiap putaran. Hal ini dimaksud untuk mengoptimalkan setiap instruksi pada CPU.
* Pipelining : adalah sebuah teknik yang memungkinkan dapat melakukan eksekusi secara simultan (sesuatu yang terjadi pada waktu bersamaan). Sehingga proses instruksi lebih efiisien.
* Large number of registers : Jumlah register yang sangat banyak. RISC di Desain dimaksudkan untuk dapat menampung jumlah register yang sangat banyak untuk mengantisipasi agar tidak terjadi interaksi yang berlebih dengan memori.

Ciri-ciri :

* Instruksi berukuran tunggal.
* Ukuran yang umum adalah 4 byte.
* Jumlah pengalamatan data sedikit.
* Tidak terdapat pengalamatan tak langsung.
* Tidak terdapat operasi yang menggabungkan operasi load/store dengan operasi aritmatika.
* Tidak terdapat lebih dari satu operand beralamat memori per instruksi.
* Tidak mendukung perataan sembarang bagi data untuk operasi load/ store.
* Jumlah maksimum pemakaian memori manajemen bagi suatu alamat data adalah sebuah instruksi.

Pengaplikasian RISC yaitu pada CPU Apple. Contoh prosessor RISC :

* Intel 960
* Itanium (IA64) dari Intel Corporation
* Alpha AXP dari DEC
* R4x00 dari MIPS Corporation.
* PowerPC dari Arsitektur POWER dari International Business Machine.
* ARM (Advanced RISC Machine dan Strong ARM

**Perbedaan Prosessor CISC dan RISC**

|  |  |
| --- | --- |
| **CISC** | **RISC** |
| Penekanan pada perangkat keras | Penekanan pada perangakat lunak |
| Instruksi kompleks multi-clock | Single clock, hanya sejumlah kecil instruksi |
| Momori-ke-memori :  “LOAD” dan “STORE” saling bekerjasama | Register-ke-register :  “LOAD” dan “STORE” adalah instruksi terpisah |
| Ukuran kode kecil, kecepatan rendah | Ukuran kode besar, kecepatan (relatif) tinggi |
| Transistor digunakan untuk menyimpan instruksi-instruksi kompleks | Transistor banyak dipakai untuk register memori |

C. Elemen – elemen instruksi

* Operation Code (OPCODE) / Kode Operasi

Kerjakan ini.

* Source Operand Reference / Alamat Asal Operand
* Terhadap isi alamat ini :
* Result Operand Reference / Alamat Hasil Operand

Letakkan hasilnya di alamat ini.

* Next Instruction Reference

Alamat yang berisi instruksi selanjutnya.

Format instruksi (biner):

Misal instruksi dengan 2 alamat operand : ADD A,B A dan B adalah suatu alamat register.

Beberapa simbolik instruksi:

|  |  |
| --- | --- |
| ADD : Add (jumlahkan)  SUB : Subtract (Kurangkan)  MPY/MUL : Multiply (Kalikan)  DIV : Divide (Bagi)  LOAD : Load data dari register/memory | STOR : Simpan data ke register/memory  MOVE : pindahkan data dari satu tempat ke tempat lain  SHR : shift kanan data  SHL : shift kiri data .dan lain-lain |

Cakupan jenis instruksi:

1. Data processing : Aritmetik (ADD, SUB, dsb); Logic (AND, OR, NOT, SHR) konversi data.
2. Data storage (memory) : Transfer data (STOR, LOAD, MOVE, dsb).
3. Data movement : Input dan Output ke modul I/O.
4. Program flow control : JUMP, HALT, dsb.

**Bentuk instruksi:**

1. Format instruksi 3 alamat

Mempunyai bentuk umum seperti : [OPCODE][AH],[AO1],[AO2]. Terdiri dari satu alamat hasil, dan dua alamat operand, misal SUB Y,A,B yang mempunyai arti dalam bentuk algoritmik : Y : = A – B dan arti dalam bentuk penjelasan : kurangkan isi reg a dengan isi reg B, kemudian simpan hasilnya di reg Y. bentuk bentuk pada format ini tidak umum digunakan di dalam komputer, tetapi tidak dimungkinkan ada penggunaanya, dalam peongoperasiannya banyak register sekaligus dan program lebih pendek.

Contoh:  
A, B, C, D, E, T, Y adalah register  
Program: Y = (A – B) / ( C + D × E)  
SUB Y, A, B Y := A – B  
MPY T, D, E T := D × E  
ADD T, T, C T := T + C  
DIV Y, Y, T Y:= Y / T  
Memerlukan 4 operasi

2. Format instruksi 2 alamat

Mempunyai bentuk umum : [OPCODE][AH],[AO]. Terdiri dari satu alamat hasil merangkap operand, satu alamat operand, missal : SUB Y,B yang mempunyai arti dalam algoritmik : Y:= Y – B dan arti dalam bentuk penjelasan : kurangkan isi reg Y dengan isi reg B, kemudian simpan hasillnya di reg Y. bentuk bentuk format ini masih digunakan di komputer sekarang, untuk mengoperasikan lebih sedikit register, tapi panjang program tidak bertambah terlalu banyak.

Contoh :  
A, B, C, D, E, T, Y adalah register  
Program: Y = (A – B) / ( C + D × E)  
MOVE Y, A Y := A  
SUB Y, B Y := Y – B  
MOVE T, D T := D  
MPY T, E T := T × E  
ADD T, C T := T + C  
DIV Y, T Y:= Y / T  
Memerlukan 6 operasi

3. Format instruksi 1 alamat

Mempunyai bentuk umum : [OPCODE][AO]. Terdiri dari satu alamat operand, hasil disimpan di accumulator, missal : SUB B yang mempunyai arti dalam algoritmik : AC:= AC – B dan arti dalam bentuk penjelasan : kurangkan isi Acc dengan isi reg B, kemudian simpan hasillnya di reg Acc. Bentuk-bentuk format ini masih digunakan di komputer jaman dahulu, untuk mengoprasikan di perlukan satu register, tapi panjang program semakin bertambah.

Contoh :  
A, B, C, D, E, Y adalah register  
Program: Y = (A – B) / ( C + D × E)  
LOAD D AC := D  
MPY E AC := AC × E  
ADD C AC := AC + C  
STOR Y Y := AC  
LOAD A AC := A  
SUB B AC := AC – B  
DIV Y AC := AC / Y  
STOR Y Y := AC  
Memerlukan 8 operasi

4. Format instruksi 0 alamat

Mempunyai bentuk umum : [OPCODE]. Terdiri dari semua alamat operand implicit, disimpan dalam bentuk stack. Operasi yang biasanya membutuhkan 2 operand, akan mengambil isi stack paling atas dan dibawahnya missal : SUB yang mempunyai arti dalam algoritmik : S[top]:=S[top-1]-S[top] dan arti dalam bentuk penjelasan : kurangkan isi stack no2 dari atas dengan isi stack paling atas, kemudian simpan hasilnya di stack paling atas, untuk mengoprasikan ada beberapa instruksi khusus stack PUSH dan POP.

Contoh :  
A, B, C, D, E, Y adalah register  
Program: Y = (A – B) / ( C + D × E)  
PUSH A S[top] := A  
PUSH B S[top] := B  
SUB S[top] := A – B  
PUSH C S[top] := C  
PUSH D S[top] := D  
PUSH E S[top] := E  
MPY S[top] := D × E  
ADD S[top] := C + S[top]  
DIV S[top] := (A – B) /S[top]  
POP Y Out := S[top]  
Memerlukan 10 operasi

**DESAIN SET INSTRUKSI**

Desain set instruksi merupakan masalah yang sangat komplek yang melibatkan banyak aspek, diantaranya adalah:

1. Kelengkapan set instruksi
2. Ortogonalitas (sifat independensi instruksi)
3. Kompatibilitas : – Source code compatibility – Object code Compatibility

Selain ketiga aspek tersebut juga melibatkan hal-hal sebagai berikut:

1. Operation Repertoire : Berapa banyak dan operasi apa saja yang disediakan, dan berapa sulit operasinya
2. Data Types : tipe/jenis data yang dapat olah Instruction Format: panjangnya, banyaknya alamat, dsb.
3. Register : Banyaknya register yang dapat digunakan 4.Addressing: Mode pengalamatan untuk operand

**FORMAT INSTRUKSI**

Suatu instruksi terdiri dari beberapa field yang sesuai dengan elemen dalam instruksi tersebut. Layout dari suatu instruksi sering disebut sebagai Format Instruksi (Instruction Format).

**OPCODE OPERAND REFERENCE OPERAND REFERENCE JENIS-JENIS OPERAND**

* Addresses (akan dibahas pada addressing modes)
* Numbers : – Integer or fixed point – Floating point – Decimal (BCD)
* Characters : – ASCII – EBCDIC
* Logical Data : Bila data berbentuk binary: 0 dan 1

**JENIS INSTRUKSI**

* Data processing: Arithmetic dan Logic Instructions
* Data storage: Memory instructions
* Data Movement: I/O instructions
* Control: Test and branch instructions

**TRANSFER DATA**

* Menetapkan lokasi operand sumber dan operand tujuan.
* Lokasi-lokasi tersebut dapat berupa memori, register atau bagian paling atas daripada stack.
* Menetapkan panjang data yang dipindahkan.
* Menetapkan mode pengalamatan.
* Tindakan CPU untuk melakukan transfer data adalah :  
  a. Memindahkan data dari satu lokasi ke lokasi lain.  
  b. Apabila memori dilibatkan :  
  1. Menetapkan alamat memori.  
  2. Menjalankan transformasi alamat memori virtual ke alamat memori aktual.  
  3. Mengawali pembacaan / penulisan memori.

Operasi set instruksi untuk transfer data :

* MOVE : memindahkan word atau blok dari sumber ke tujuan
* STORE : memindahkan word dari prosesor ke memori.
* LOAD : memindahkan word dari memori ke prosesor.
* EXCHANGE : menukar isi sumber ke tujuan.
* CLEAR / RESET : memindahkan word 0 ke tujuan.
* SET : memindahkan word 1 ke tujuan.
* PUSH : memindahkan word dari sumber ke bagian paling atas stack.
* POP : memindahkan word dari bagian paling atas sumber

**ARITHMETIC**

Tindakan CPU untuk melakukan operasi arithmetic :

1. Transfer data sebelum atau sesudah.
2. Melakukan fungsi dalam ALU.
3. Menset kode-kode kondisi dan flag.

Operasi set instruksi untuk arithmetic :

|  |  |
| --- | --- |
| 1. ADD : penjumlahan 2. SUBTRACT : pengurangan 3. MULTIPLY : perkalian 4. DIVIDE : pembagian | 5. ABSOLUTE  6. NEGATIVE  7. DECREMENT  8. INCREMENT |

Operasi set instruksi untuk operasi logical :

|  |
| --- |
| 1. AND, OR, NOT, EXOR 2. COMPARE : melakukan perbandingan logika. 3. TEST : menguji kondisi tertentu. 4. SHIFT : operand menggeser ke kiri atau kanan menyebabkan konstanta pada ujung bit. 5. ROTATE : operand menggeser ke kiri atau ke kanan dengan ujung yang terjalin. |

**CONVERSI**

1. Tindakan CPU sama dengan arithmetic dan logical.
2. Instruksi yang mengubah format instruksi yang beroperasi terhadap format data.
3. Misalnya pengubahan bilangan desimal menjadi bilangan biner.
4. Operasi set instruksi untuk conversi :

TRANSLATE : menterjemahkan nilai-nilai dalam suatu bagian memori berdasrkan tabel korespodensi.  
CONVERT : mengkonversi isi suatu word dari suatu bentuk ke bentuk lainnya.

**INPUT / OUPUT**

* Tindakan CPU untuk melakukan INPUT /OUTPUT :

1. Apabila memory mapped I/O maka menentukan alamat memory mapped.

2. Mengawali perintah ke modul I/O

* Operasi set instruksi Input / Ouput :

1. INPUT : memindahkan data dari pernagkat I/O tertentu ke tujuan

2. OUTPUT : memindahkan data dari sumber tertentu ke perangkat I/O

3. START I/O : memindahkan instruksi ke prosesor I/O untuk mengawali operasi I/O

4. TEST I/O : memindahkan informasi dari sistem I/O ke tujuan TRANSFER CONTROL

* Tindakan CPU untuk transfer control : Mengupdate program counter untuk subrutin , call / return.
* Operasi set instruksi untuk transfer control :

1. JUMP (cabang) : pemindahan tidak bersyarat dan memuat PC dengan alamat tertentu.  
2. JUMP BERSYARAT : menguji persyaratan tertentu dan memuat PC dengan alamat tertentu atau tidak melakukan apa tergantung dari persyaratan.

3. JUMP SUBRUTIN : melompat ke alamat tertentu.

4. RETURN : mengganti isi PC dan register lainnya yang berasal dari lokasi tertentu.

5. EXECUTE : mengambil operand dari lokasi tertentu dan mengeksekusi sebagai instruksi

6. SKIP : menambah PC sehingga melompati instruksi berikutnya.

7. SKIP BERSYARAT : melompat atau tidak melakukan apa-apa berdasarkan pada persyaratan.

8. HALT : menghentikan eksekusi program.

9. WAIT (HOLD) : melanjutkan eksekusi pada saat persyaratan dipenuhi

10. NO OPERATION : tidak ada operasi yang dilakukan.

**CONTROL SYSTEM**

Hanya dapat dieksekusi ketika prosesor berada dalam keadaan khusus tertentu atau sedang mengeksekusi suatu program yang berada dalam area khusus, biasanya digunakan dalam sistem operasi. \* Contoh : membaca atau mengubah register kontrol.

**JUMLAH ALAMAT (NUMBER OF ADDRESSES)**

Salah satu cara tradisional untuk menggambarkan arsitektur prosessor adalah dengan melihat jumlah alamat yang terkandung dalam setiap instruksinya.

Jumlah alamat maksimum yang mungkin diperlukan dalam sebuah instruksi :

1. Empat Alamat ( dua operand, satu hasil, satu untuk alamat instruksi berikutnya)

2. Tiga Alamat (dua operand, satu hasil)

3. Dua Alamat (satu operand merangkap hasil, satunya lagi operand)

4. Satu Alamat (menggunakan accumulator untuk menyimpan operand dan hasilnya)