**Pengantar Organisasi dan Arsitektur Komputer**

“ Set Instruksi ”

# Oleh : Farhat, ST, MMSI, MSc

**{ Diolah dari berbagai Sumber }**

# SET INSTRUKSI DAN PENGALAMATAN

Set instruksi (instruction set) adalah sekumpulan lengkap instruksi yang dapat di mengerti oleh sebuah CPU, set instruksi sering juga disebut sebagai bahasa mesin (machine code), karna aslinya juga berbentuk biner kemudian dimengerti sebagai bahasa assembly, untuk konsumsi manusia (programmer), biasanya digunakan representasi yang lebih mudah dimengerti oleh manusia.

Sebuah instruksi terdiri dari sebuah opcode, biasanya bersama dengan beberapa informasi tambahan seperti darimana asal operand-operand dan kemana hasil-hasil akan ditempatkan. Subyek umum untuk menspesifikasikan di mana operand-operand berada (yaitu, alamat-alamatnya) disebut pengalamatan

Pada beberapa mesin, semua instruksi memiliki panjang yang sama, pada mesin-mesin yang lain mungkin terdapat banyak panjang berbeda. Instruksi-instruksi mungkin lebih pendek dari, memiliki panjang yang sama seperti, atau lebih panjang dari panjang word. Membuat semua instruksi memiliki panjang yang sama lebih muda dilakukan dan membuat pengkodean lebih mudah tetapi sering memboroskan ruang, karena semua instruksi dengan demikian harus sama panjang seperti instruksi yang paling panjang.

Di dalam sebuah instruksi terdapat beberapa elemen-elemen instruksi:

1. Operation code (op code)
2. Source operand reference
3. Result operand reference
4. Xext instruction preference

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

Beberapa simbolik instruksi:

1. ADD : Add (jumlahkan)
2. SUB : Subtract (Kurangkan)
3. MPY/MUL : Multiply (Kalikan)
4. DIV : Divide (Bagi)
5. LOAD : Load data dari register/memory
6. STOR : Simpan data ke register/memory
7. MOVE : Pindahkan data dari satu tempat ke tempat lain
8. SHR : Shift kanan data
9. SHL : Shift kiri data .dan lain-lain

## CAKUPAN JENIS INSTRUKSI

1. Data processing : Aritmetik (ADD, SUB, dsb); Logic (AND, OR, NOT, SHR, dsb); 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

- **Format instruksi 3 alamat**

Mempunyai bentuk umum seperti : [OPCODE][AH],[AO1],[AO2]. Terdiri dari satu alamt 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 computer, tetapi tidak dimungkinkan ada pengunaanya, dalam peongoprasianya 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**

## - 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 computer sekarang, untuk mengoprasikan 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**

## - 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 computer 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**

## - 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**

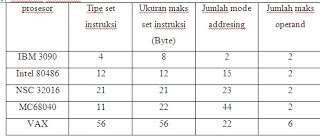
**ADD S[top] := C + S[top]**

**DIV S[top] := (A - B) /S[top]**

**POP Y Out := S[top]**

**Memerlukan 10 operasi**

## SET INSTRUKSI PADA CISC

Berikut ini merupakan karakteristik set instruksi yang digunakan pada beberapa computer yang memiliki arsitektur CISC

## PERBANDINGAN SET INSTRUKSI

Beberapa computer CISC (Complex Instruction Set Computer) menggunakan cara implist dalam menentukan mode addressing pada setiap set instruksinya. Penentuan mode addressing dengan cara implicit memiliki arti bahwa pada set instruksi tidak di ada bagian yang menyatakan tipe dari mode addressing yang digunakan, deklarasi dari mode addressing itu berada menyatu dengan opcode. Lain hal nya dengan cara imsplisit, cara eksplisit sengaja menyediakan tempat pada set instruksi untuk mendeklarasikan tipe mode addressing. Pada cara eksplisit deklarasi opcode dan mode addressing berada terpisah.

Data pada tempat deklarasi mode addressing diperoleh dari logaritma basis dua jumlah mode addressing. Jika deklarasi mode addressing dilakukan secara implicit akan menghemat tempat dalam set instruksi paling tidak satu bit untuk IBM 3090 dan 6 bit untuk MC68040. Perubahan satu bit pada set instruksi akan memberikan jangkauan alamat memori lebih luas mengingat range memori dinyatakan oleh bilangan berpangkat dua.

## IMPLEMENTASI HARDWARE

Setiap set instruksi yang berbeda membutuhkan perangkat hardware yang berbeda pula. Hal ini terjadi karena set instruksi yang berbeda menyimpan informasi yang berbeda sehingga dibutuuhkan hardware yang berbeda untuk mengubah set instruksi tersebut ke bentuk sinyal- sinyal control.

Untuk mendapatkan opcode berikutnya prosesor harus mengetahui letak dari opcode tersebut secara pasti pada memory. Karena tipe dari mode addressing sangat mempengaruhi posisi dari operand, maka secara tidak langsung mode addressing mempengaruhi letak opcode selanjutnya. Sehingga dapat disimpulkan kedua cara pendeklarasian mode addressing tersebut turut mempengaruhi arsitektur hardware dari computer.

## PENGALAMATAN

Program biasanya ditulis dalam bahasa tingkat tinggi, yang memunkinkan program untuk menggunakan konstanta, variable local dan global, pointer, dan array. Pada saat mentranslasi program bahsa tingkat tinggi menjadi bahsa assembly, compiler harus mampu mengimplimentasi konstruksi ini menggunakan fasilitas yang disediakan dalam set instruksi computer dimana program akan dijalankan. cara yang berbeda dalam menentukan lokasi suatu operand ditetapkan dalam suatu instruksi yang disebut sebagai mode pengalamatan.

## IMPLEMENTASI VARIABLE DAN KONSTANTA

Variable dan konstanta adalah tipe data yang paling sederhana dan terdapat dalam hampir setiap program computer. Dalam bahasa assembley, suatu variable dinyatakan dengan mengalokasikan suatu register atau lokasi memori untuk menyimpan nilainya. Sehingga nilai tersebut dapat diubah seperlunya menggunakan instruksi sesuai.

Kita mengakses operand dengan menetapkan nama register atau alamat lokasi memori tempat operand berada. Definisi yang presisi dari dua mode tersebut adalah:

1. Mode register operand adalah isi register prosesor, nama alamat register dinyatakan dalam instruksi tersebut.
2. Mode absolute operand adalah lokasi memori, alamat lokasi dinyatakan secara eksplisit didalam instruksi tersebut. (pada bebrapa bahasa assembly, mode ini disebut direct).
3. Mode immediate operand dinyatakan secara eksplisit dalam instruksi, misalnya, instruksi Move 200 immediete, RO.

## INDIRECTION DAN POINTER

Pada mode pengalamatan berikutnya, instruksi tidak menyatakana operand atau alamatnya secara eksplisit. Sebaliknya, instruksi menyediakan informasi dari nama alamat memori suatu operand dapat ditetapkan. Kita menyebut alamat ini sebagai effective address (EA) suatu operand.

Selain kesederhanaanya yang tampak jelas, pengalamatan indirect melalui memori telah terbukti memiliki keterbatasan pengunaan sebagai mode pengalamatan, dan jarang di gunakan dalam computer modern, pengalamatan indirect melalui register digunakan secara luas. Maka pengalamatan indirect melalui register memungkinkan untuk mengakses variable global dengan terlebih dahulu me-load alamat operand dalam suatu register.

Secara umum teknik addressing yang sering dilakukan adalah:

## Immediate addressing

Operand (data yang akan dikomputasi) berada langsung pada set instruksi.

## Direct Addressing

Operand berada pada memori, set instruksi memegang alamat lokasi memori dimana operand tersebut berada.

## Indirect Addresing

Operand berada pada memori, untuk mendapatkan operand ini CPU harus melakukan penelusuran dua kali yaitu dari data alamat memori yang ada pada set instruksi serta alamat yang ditunjuk oleh alamat memori yang diperoleh dari set instruksi tadi.

## Register addressing

Operand berada pada register, cara kerjanya mirip dengan direct addressing hanya saja CPU mengakses alamat register bukan alamat memori.

## Register Indirect Addressing

Operand berada pada memori, untuk mendapatkan operand CPU harus mengakses register terlebih dahulu karena informasi lokasi operand berada pada register.

## Displacement

Operand berada pada memori, cara kerjanya merupakan gabungan dari teknik direct addressing dan register indirect addressing.

## Stack

Operand berada pada stack, operand secara berkala dimasukan ke stack sehingga ketika operand dibutuhkan maka operand sudah berada pada “top of the stack”.

Teknik pengalamatan tersebut harus dapat memenuhi kebutuhan komputasi yang dilakukan oleh computer yang secara garis besar dapat dibagi kedalam tiga kategori yaitu:

* Operasi load (memasukan data).
* Operasi branch (percabangan).
* Operasi aritmatik dan logika.



**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*).

**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

1. Data processing: *Arithmetic dan Logic Instructions*
2. Data storage: *Memory instructions*
3. Data Movement: *I/O instructions*
4. 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 :

* 1. Memindahkan data dari satu lokasi ke lokasi lain.
  2. Apabila memori dilibatkan :
     + Menetapkan alamat memori.
     + Menjalankan transformasi alamat memori virtual ke alamat memori aktual.
     + 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 5. ABSOLUTE
  2. SUBTRACT : pengurangan 6. NEGATIVE

1. MULTIPLY : perkalian 7. DECREMENT
2. DIVIDE : pembagian 8. INCREMENT Nomor 5 sampai 8 merupakan instruksi operand tunggal.

## LOGICAL

* + Tindakan CPU sama dengan arithmetic

## 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.

## KONVERSI

* Tindakan CPU sama dengan arithmetic dan logical.
* Instruksi yang mengubah format instruksi yang beroperasi terhadap format data.
* Misalnya pengubahan bilangan desimal menjadi bilangan biner.

## OPERASI SET INSTRUKSI UNTUK KONVERSI :

1. TRANSLATE : menterjemahkan nilai-nilai dalam suatu bagian memori berdasrkan tabel korespodensi.
2. 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 danmemuat 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 menyimpancoperand dan hasilnya)

## MACAM-MACAM INSTRUKSI MENURUT JUMLAH OPERASI YANG DISPESIFIKASIKAN

1. Memori To Register Instruction
2. Memori To Memori Instruction
3. Register To Register Instruction

## ADDRESSING MODES

Jenis-jenis addressing modes (Teknik Pengalama-tan) yang paling umum:

* + *Immediate*
  + *Direct*
  + *Indirect*
  + *Register*
  + *Register Indirect*
  + *Displacement*
  + *Stack*

## GAMBAR ADDRESSING MODE

