**Set Instruksi Program dan Jenis-jenis Instruksi**

**Pengertian Set Intruksi**

Set Instruksi (bahasa Inggris: *Instruction Set*, atau *Instruction Set Architecture* (ISA)) didefinisikan sebagai suatu aspek dalam arsitektur komputer yang dapat dilihat oleh para pemrogram. Secara umum, ISA ini mencakup jenis data yang didukung, jenis instruksi yang dipakai, jenis register, mode pengalamatan, arsitektur memori, penanganan interupsi, eksepsi, dan operasi I/O eksternalnya (jika ada).

ISA merupakan sebuah spesifikasi dari Pullman semua kode-kode biner (opcode) yang diimplementasikan dalam bentuk aslinya (native form) dalam sebuah desain prosesor tertentu. Kumpulan opcode tersebut, umumnya disebut sebagai bahasa mesin (machine language) untuk ISA yang bersangkutan. ISA yang populer digunakan adalah set instruksi untuk chip Intel x86, IA-64, IBM PowerPC, Motorola 68000, Sun SPARC, DEC Alpha, dan lain-lain.

ISA kadang-kadang digunakan untuk membedakan kumpulan karakteristik yang disebut di atas dengan mikroarsitektur prosesor, yang merupakan kumpulan teknik desain prosesor untuk mengimplementasikan set instruksi (mencakup microcode, pipeline, sistem cache, manajemen daya, dan lainnya). Komputer-komputer dengan mikroarsitektur berbeda dapat saling berbagi set instruksi yang sama. Sebagai contoh, prosesor Intel Pentium dan prosesor AMD Athlon mengimplementasikan versi yang hampir identik dari set instruksi Intel x86, tetapi jika ditinjau dari desain internalnya, perbedaannya sangat radikal. Konsep ini dapat diperluas untuk ISA-ISA yang unik seperti TIMI yang terdapat dalam IBM System/38 dan IBM IAS/400. TIMI merupakan sebuah ISA yang diimplementasikan sebagai perangkat lunak level rendah yang berfungsi sebagai mesin virtual. TIMI didesain untuk meningkatkan masa hidup sebuah platform dan aplikasi yang ditulis untuknya, sehingga mengizinkan platform tersebut agar dapat dipindahkan ke perangkat keras yang sama sekali berbeda tanpa harus memodifikasi perangkat lunak (kecuali yang berkaitan dengan TIMI). Hal ini membuat IBM dapat memindahkan platform AS/400 dari arsitektur mikroprosesor CISC ke arsitektur mikroprosesor POWER tanpa harus menulis ulang bagian-bagian dari dalam sistem operasi atau perangkat lunak yang diasosiasikan dengannya.

Ketika mendesain mikroarsitektur, para desainer menggunakan Register Transfer Language (RTL) untuk mendefinisikan operasi dari setiap instruksi yang terdapat dalam ISA.

Sebuah ISA juga dapat diemulasikan dalam bentuk perangkat lunak oleh sebuah interpreter. Karena terjadi translasi tambahan yang dibutuhkan untuk melakukan emulasi, hal ini memang menjadikannya lebih lambat jika dibandingkan dengan menjalankan program secara langsung di atas perangkat keras yang mengimplementasikan ISA tersebut. Akhir-akhir ini, banyak vendor ISA atau mikroarsitektur yang baru membuat perangkat lunak emulator yang dapat digunakan oleh para pengembang perangkat lunak sebelum implementasi dalam bentuk perangkat keras dirilis oleh vendor.

**Set Intruksi Mode dan Jenis Instruksi**

1. Format-Format Instruksi

Format instruksi menentukan layout bit di dalam suatu instruksi yang mencakup opcode, dan nol operand atau lebih

1.1. Representasi Instruksi

Instruksi direpresentasikan oleh sekumpulan bit. Instruksi dibagi menjadi beberapa field, di mana field-field ini berkaitan erat dengan elemen-elemen yang mengisi instruks

1.2. Panjang Instruksi

Panjang instruksi harus merupakan kelipatan panjang karakter, yang umumnya 8 bit dan kelipatan panjang bilangan fixed point. Untuk memahami hal ini, kita harus menggunakan istilah word. Umumnya ukuran word menentukan ukuran bilangan-bilangan fixed point dan berhubungan dengan ukuran transfer memori.

1.3. Alokasi Bit

**Yang berkaitan dengan penggunaan bit-bit pengalamatan :**

1. Jumlah mode pengalamatan dan operand
2. Jumlah set register
3. Register dan Memori
4. Jangkauan Alamat

1.4 Jenis-Jenis Instruksi

**Jenis-jenis instruksi dapat digolongkan atas :**

1. Data Processing : instruksi-instruksi aritmatika dan logika
2. Data Storage : instruksi-instruksi memori
3. Data Movement : instruksi I/O
4. Control : instruksi pemeriksaan dan pencabangan

Instruksi aritmetika (arithmetic instruction) memiliki kemampuan untuk mengolah data numeric. Sedangkan instruksi logika (logic instruction) beroperasi pada bit-bit word sebagai bit, bukan sebagai bilangan. Operasi-operasi tersebut dilakukan terutama dilakukan untuk data di register CPU. Instruksi-inslruksi memori diperlukan untuk memindah data yang terdapat di memori dan register. Instruksi-instruksi I/O diperlukan untuk memindahkan program dan data kedalam memori dan mengembalikan hasil komputasi kepada pengguna. Instruksi-instruksi control digunakan untuk memeriksa nilai data, status komputasi dan mencabangkan ke set instruksi lain.

**1.5 Rancangan Set Instruksi**

Yang menjadi permasalahan dalam rancangan set instruksi adalah

1. Operation Repertoire
2. Data Types
3. Instruction Format
4. Registers
5. Addressing

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):**

Missal 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:**

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

Data storage (memory)  : Transfer data (STOR, LOAD, MOVE, dsb)

Data movement              : Input dan Output ke modul I/O

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.

**ELEMEN-ELEMEN DARI INSTRUKSI MESIN (SET INSTRUKSI)**

\* Operation Code (opcode) : menentukan operasi yang akan dilaksanakan

\* Source Operand Reference : merupakan input bagi operasi yang akan dilaksanakan

\* Result Operand Reference : merupakan hasil dari operasi yang dilaksanakan

\* Next instruction Reference : memberitahu CPU untuk mengambil (fetch) instruksi berikutnya setelah instruksi yang dijalankan selesai. Source dan result operands dapat berupa salah satu diantara tiga jenis berikut ini:

* Main or Virtual Memory
* CPU Register
* I/O Device

**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 5. ABSOLUTE  
2. SUBTRACT : pengurangan 6. NEGATIVE  
3. MULTIPLY : perkalian 7. DECREMENT  
4. 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.

**CONVERSI**

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 conversi :  
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 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)

Macam-macam instruksi menurut jumlah operasi yang dispesifikasikan  
1. O – Address Instruction  
2. 1 – Addreess Instruction.  
3. N – Address Instruction  
4. M + N – Address Instruction

Macam-macam instruksi menurut sifat akses terhadap memori atau register  
1. Memori To Register Instruction  
2. Memori To Memori Instruction  
3. Register To Register Instruction

**ADDRESSING MODES**

Jenis-jenis addressing modes (Teknik Pengalamatan) yang paling umum:  
\* Immediate  
\* Direct  
\* Indirect  
\* Register  
\* Register Indirect  
\* Displacement  
\* Stack