***7***

|  |  |  |
| --- | --- | --- |
| **Mata Kuliah** | **:** | **Arsitektur dan Organisasi Komputer** |
| **Bobot Sks** | **:** | **3 sks** |
| **Dosen Pengembang** | **:** | **Catur Nugroho, S.Kom., M.Kom** |
| **Tutor** | **:** |  |
| **Capaian Pembelajaran Mata Kuliah** | **:** | Mahasiswa mampu menggunakan dan menerapkan konsep & definisi set instruksi. |
| **Kompetentsi Akhir Di Setiap Tahap (Sub-Cpmk)** | **:** | Mahasiswa mampu menggunakan dan menerapkan konsep & definisi set instruksi. |
| **Minggu Perkuliahan Online Ke-** | **:** | **Sesi 11** |

1. SET INSTRUKSI

**Apa itu set instruksi?**

Set instruksi (*instruction set*) merupakan sekumpulan lengkap instruksi yang dapat dimengerti oleh sebuah CPU, instruksinya berbentuk machine code (bahasa mesin), yang seluruhnya dalam bentuk bilangan biner.

Programmer biasanya menggunakan representasi dan bahasa yang lebih mudah dimengerti manusia, yang dikenal dengan bahasa *Assembly*.

1. KARAKTERISTIK DAN FUNGSI SET INSTRUKSI

Operasi dari CPU ditentukan oleh instruksi-instruksi yang dilaksanakan atau dijalankannya. Instruksi ini sering disebut sebagai instruksi mesin (*machine instructions*).

1. ELEMEN-ELEMEN DARI INSTRUKSI MESIN

Adapun elemen – elemen dari instruksi mesin sebagai berikut :

• *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.

• Sebuah instruksi tidak harus memiliki semua elemen tersebut, tergantung kebutuhan dan jenis instruksinya

1. KARAKTERISTIK DAN FUNGSI SET INSTRUKSI

Desain set instruksi merupakan masalah yang sangat kompleks yang melibatkan banyak aspek, di antaranya adalah :

* + 1. Kelengkapan set instruksi
    2. Ortogonalitas (sifat independensi instruksi)
    3. Kompatibilitas
* *Source code compatibility*
* *Object code compatibility*

Selain ketiga aspek tersebut, desain set instruksi juga melibatkan hal-hal sebagai berikut :

* Operation Repertoire : Berapa banyak operasi yang disediakan, operasi apa saja yang disediakan, dan berapa sulit operasinya.
* *Data Types* : Tipe atau jenis data apa saja yang dapat diolah.
* *Instruction Format* : Berapa panjang instruksinya, berapa banyak alamatnya, dan sebagainya
* *Register* : Banyaknya register yang dapat digunakan
* *Addressing* : Mode pengalamatan untuk operand.

1. **PENYIMPANAN OPERAND**

Semua instruksi yang akan dieksekusi terletak di dalam CPU, Sebuah operasi hanya membutuhkan register sebagai tempat membaca/menyimpan operand sementara, Adakalanya operand disimpan di lokasi lain melalui register yang berisi alamat tempat penyimpanan tersebut (memory, cache, modul I/O).

1. **REPRESENTASI INSTRUKSI**

Pada bahasa mesin, setiap instruksi berbentuk pola bit biner yang unik. Agar dapat dimengerti manusia, dibuatlah representasi simbolik instruksi, biasanya berupa singkatan yang disebut dengan mnemonic.

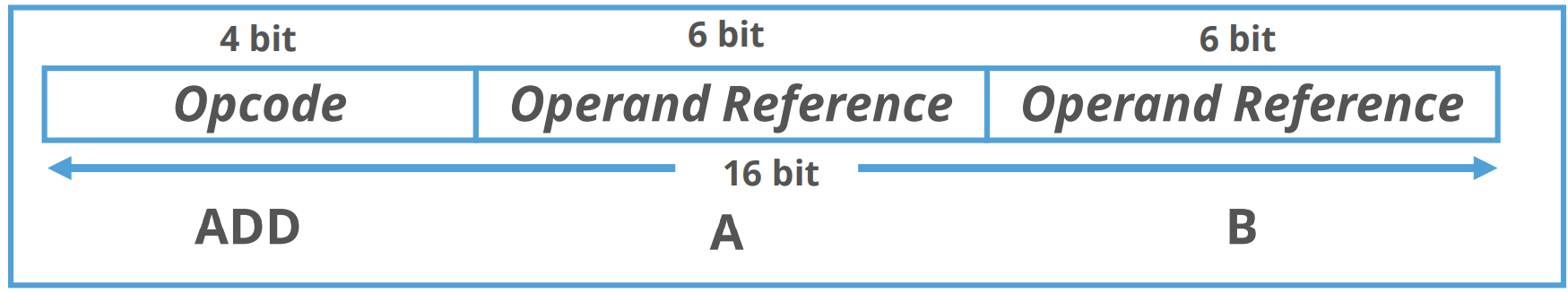
**Contoh: ADD, SUB, LOAD.**

Sedangkan alamat operand direpresentasikan sebagai berikut: ADD A, B (menambahkan operand pada posisi A dan B).

Ada beberapa jenis representasi instruksi, yang dibedakan oleh jumlah alamat operand yang dapat diterima oleh satu baris instruksi.

1. **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).



( Disimpan dalam bentuk biner )

1. CONTOH SET INSTRUKSI

Desain set instruksi merupakan masalah yang sangat kompleks yang melibatkan banyak aspek, di antaranya adalah :

* ADD : *add* (menjumlahkan/menambahkan data)
* SUB : *subtract* (mengurangkan data)
* MPY/MUL : *multiply* (mengalikan data)
* DIV : *divide* (membagi data)
* LOAD : memuat data dari register/memory
* STOR : menyimpan data ke register/memory
* MOVE : memindahkan data dari satu tempat ke tempat lain
* SHR : shift right (menggeser data ke arah kanan)
* SHL : shift left (menggeser data ke arah kiri)

1. JENIS INSTRUKSI

* *Data Processing: arithmetic and logic instructions*
* *Data Storage: memory instructions*
* *Data Movement: I/O instructions*
* *Control: test and branch instructions*

1. **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 (tumpukan)
* POP: memindahkan word dari bagian paling atas sumber.

1. **ARITHMETIC (ARITMETIKA)**

Merupakan tindakan CPU untuk melakukan operasi aritmetika : Transfer data sebelum atau sesudah Melakukan fungsi ALU Melakukan pengaturan kode-kode kondisi dan flag Operasi set instruksi untuk aritmetika:

* ADD: penjumlahan
* SUBTRACT: pengurangan
* MULTIPLY: perkalian
* DIVIDE : pembagian
* NEGATIVE
* ABSOLUTE
* DECREMENT
* INCREMENT

1. **LOGICAL**

Merupakan Tindakan CPU untuk melakukan operasi logical sama dengan arithmetic (aritmetika). Operasi set instruksi untuk operasi *logical* :

**AND, OR, NOT, EXOR**

***COMPARE :*** Melakukan perbandingan logika.

***TEST :*** Menguji kondisi tertentu.

***SHIFT :*** Operand menggeser data ke kiri atau kanan dengan konstanta pada ujung bit

**ROTATE** : Operand menggeser ke kiri atau ke kanan dengan ujung yang terhubung (terjalin).

1. **CONVERSION**

Tindakan CPU sama dengan arithmetic dan logical, instruksi yang mengubah format instruksi yang beroperasi terhadap data.

Misalnya pengubahan bilangan desimal menjadi bilangan biner

Operasi set instruksi untuk conversion:

***TRANSLATE :*** menerjemahkan nilai-nilai dalam suatu bagian memori berdasarkan tabel korespondensi.

***CONVERT :*** mengonversi isi suatu word dari suatu bentuk ke bentuk lainnya.

1. **INPUT/OUPUT**

Tindakan CPU untuk melakukan INPUT/OUTPUT:

* Apabila memory mapped untuk I/O, maka CPU menentukan alamat *memory mapped*
* Mengawali perintah ke modul I/O

Operasi set instruksi Input/Ouput:

* INPUT: memindahkan data dari perangkat I/O tertentu ke tujuan
* OUTPUT: memindahkan data dari sumber tertentu ke perangkat I/O
* START I/O: memindahk instruksi ke prosesor I/O untuk mengawali operasi I/O
* TEST I/O: memindahkan informasi dari sistem I/O ke tujuan.

1. ADDRESSING

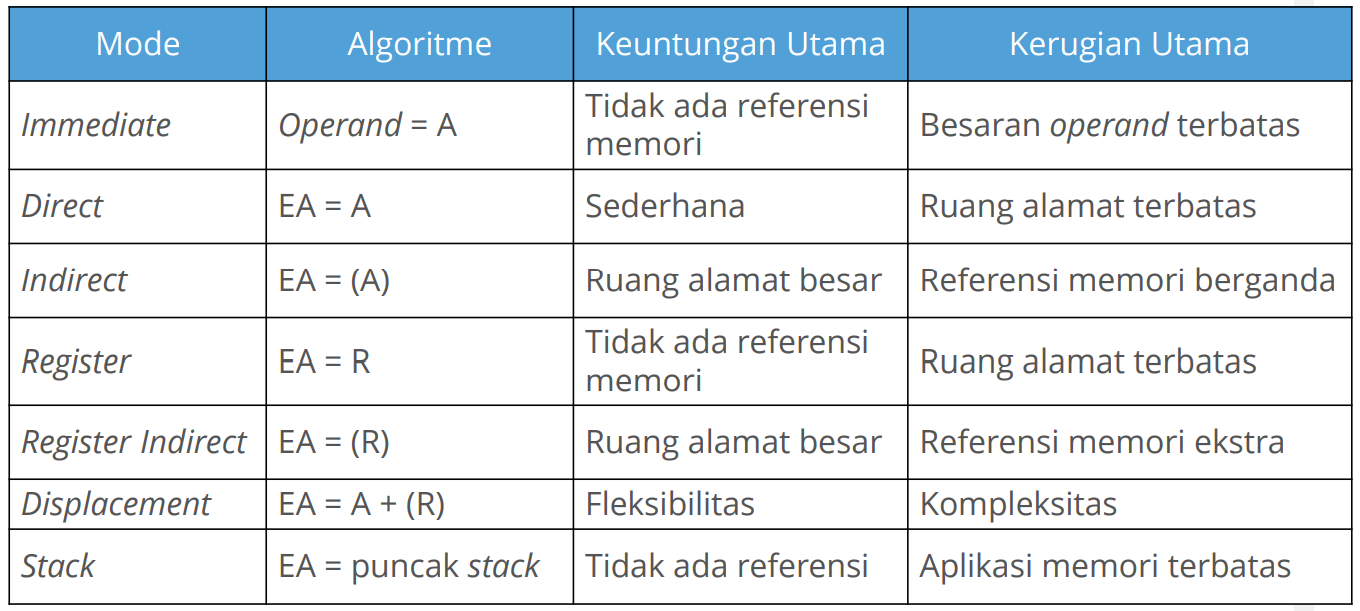
Secara umum set instruksi mencakup

1. Jenis data yang didukung
2. Jenis instruksi yang dipakai;
3. Jenis register;
4. *Mode addressing*
5. Arsitektur memori;
6. Penanganan interupsi dan eksepsi;
7. Operasi I/O eksternalnya (jika ada).

*Mode addressing* : Kumpulan instruksi menyediakan cara yang bervariasi untuk menemukan lokasi memori, cara-cara ini disebut addressing hal dapat memudahkan pemrosesan list dan pengacuan struktur data yang kompleks. Kumpulan instruksi menyediakan cara yang bervariasi untuk menemukan lokasi memori. Cara-cara ini disebut addressing. Cara ini dapat memudahkan pemrosesan list dan pengacuan struktur data yang kompleks.

Tujuan *addressing* Mengatasi keterbatasan format instruksi dan mereferensi lokasi memori yang besar. Pada setiap prosesor menggunakan mode *addressing* yang berbeda-beda; hal ini dapat menjadi pertimbangan dalam pemilihan mode addressing. Oleh karenanya kita harus mengetahui jenis prosesor yang digunakan.

*Addressing* terdiri dari tujuh mode, yaitu perbandinganya ditunjukan pada tabel



Adapunpenjelasanya adalah sebagai berikut :

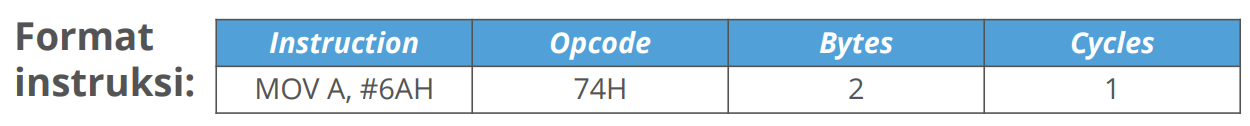
1. *Immediate addressing* : Mode ini sangat umum dipakai karena value (nilai) yang akan disimpan dalam memori langsung mengikuti kode operasi dalam memori. Mode digunakan ketika suatu value atau konstanta diketahui saat program dibuat dan tidak akan diubah selama program dieksekusi.

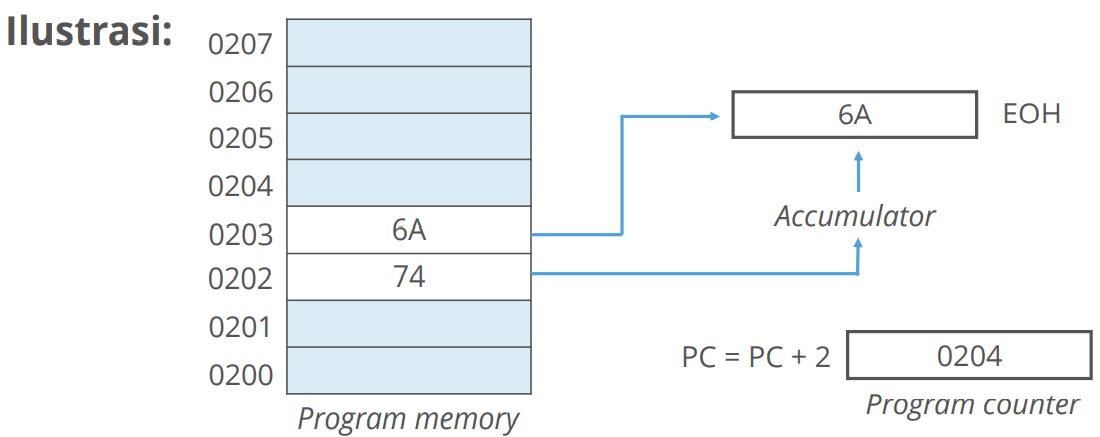
Operasi dengan mode ini membutuhkan 2-byte instruksi:

* Kode operasi
* Data

Dengan kata lain, tidak diperlukan pengambilan value dari alamat lain untuk disimpan.

Format Instruksi & ilustrasi *immediate addressing.*





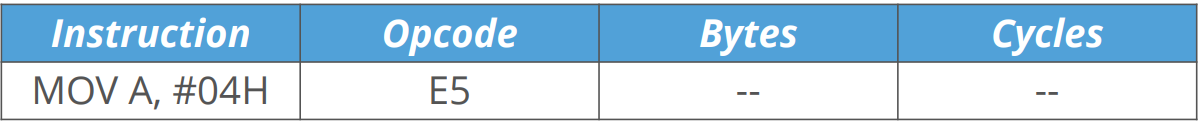
1. *Direct addressing*; Direct addressing merupakan mode addressing untuk mentransfer data antarmemori dan register.

Format Instruksi & ilustrasi *direct addressing*

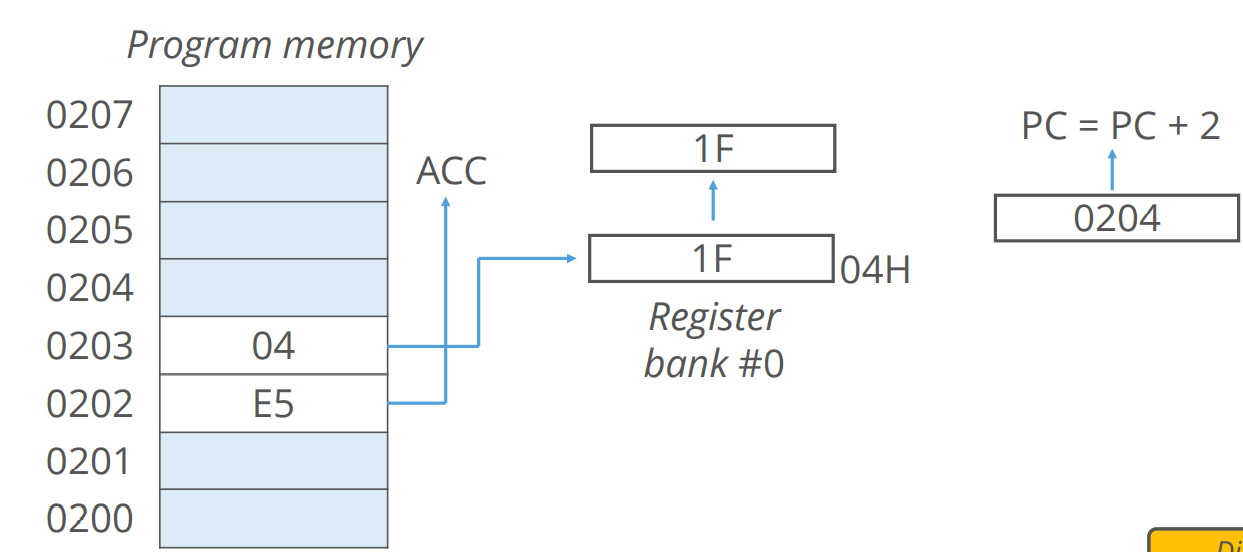
**Contoh:**

**MOV A, 04H**

**Catatan: 04H Merupakan alamat memori tempat data disimpan.**



ilustrasi *direct addressing*



1. *Indirect addressing* Indirect addressing merupakan mode addressing yang digunakan untuk mentransfer data/byte/word antarregister dan lokasi yang alamatnya ditunjukkan oleh isi suatu register.Format Instruksi & ilustrasi *indirect addressing*

*Contoh:*

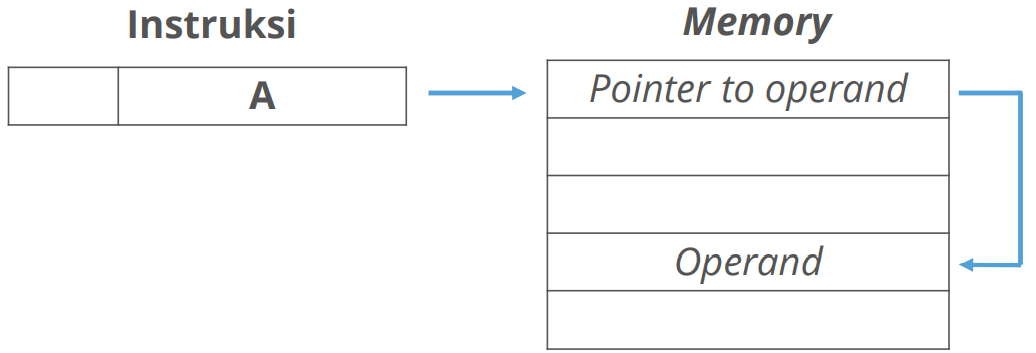
* ***ADD [A]***

keterangan: tambahkan isi memori yang ditunjuk oleh isi alamat A ke akumulator.

* ***MOV [BX], AX***

keterangan: pindahkan isi register AX ke dalam alamat memori yang ditunjukkan oleh isi register BX.

ilustrasi *indirect addressing*



1. *Register addressing* Mode register addressing mirip dengan direct addressing.

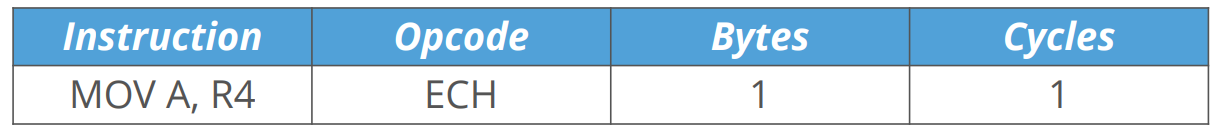
Perbedaannya terletak pada field alamat yang mengacu pada register, bukan pada memori utama, addressing antarregister ini harus digunakan dengan register yang berukuran sama, seperti AL dan BH, CX dan AX.

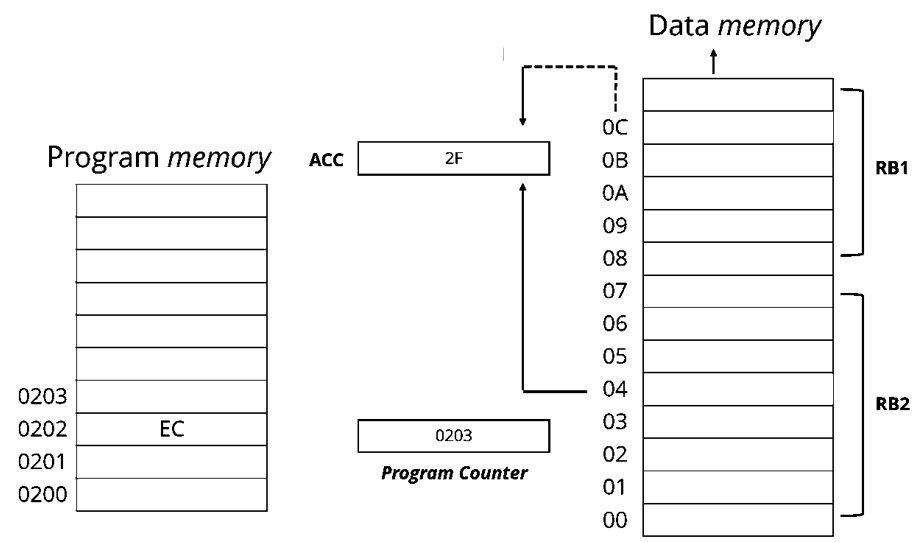
Format Instruksi & ilustrasi register addressing

Contoh:

* MOV AX, CX
* MOV A, R4

keterangan: pindahkan isi register R4 dan akumulator..

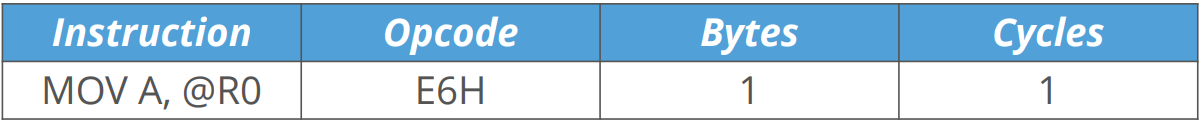


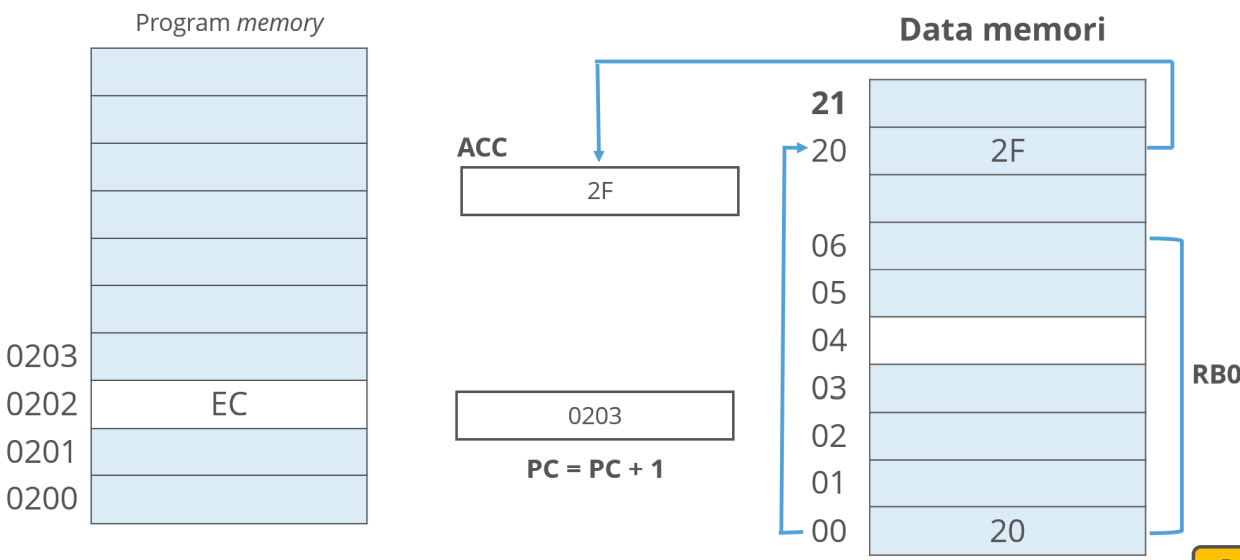


1. Register indirect addressing Mode register indirect addressing mirip dengan mode indirect addressing, perbedaannya terletak pada lokasi field alamatnya (alamat register). Letak operand berada pada memori yang ditunjuk oleh isi register.Format Instruksi & ilustrasi register addressing

Contoh:

**MOV A, @R0**

* R0 merupakan alamat yang digunakan untuk menampung data yang akan dikirim ke akumulator.
* Jika R0 = 20H terdapat data 2FH di alamat 20H maka perintah ini adalah mengirimkan nilai 2FH ke akumulator.

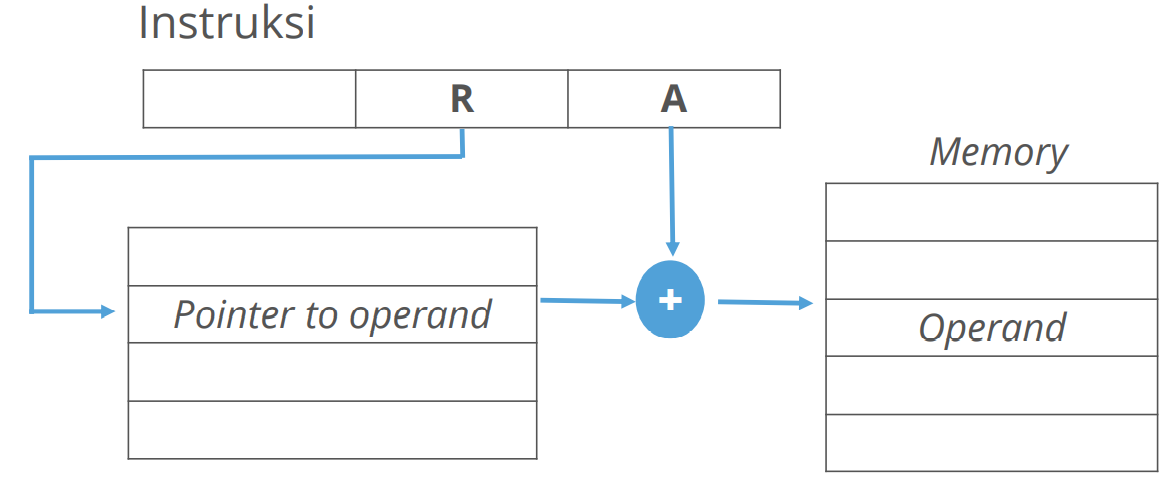


1. *Displacement addressing* yaitu menggabungkan kemampuan *direct addressing* dan register indirect addressing. Mode ini mensyaratkan instruksi memiliki dua buah field alamat, yaitu eksplisit dan implisit, field eksplisit bernilai A dan field implisit mengarah pada register.

Contoh:

**MOV A, @R0**

* Operand berada pada alamat A ditambah isi register.
* Tiga model *displacement addressing*:
  + *Relative addressing;*
  + *Base register addressing;*
  + *Indexing.*



1. Stack addressing Stack memiliki prinsip *Last In First Out* (LIFO), pada stack, pointer berada paling atas. Dua elemen teratas stack dapat berada di dalam register CPU yang artinya stack pointer mereferensi ke elemen stack ke-3, pointer stack tetap berada di dalam register.
2. Mode addressing X86 & ARM

Mekanisme translasi alamat untuk menghasilkan sebuah alamat dinamakan *virtual address* atau *effective address.*

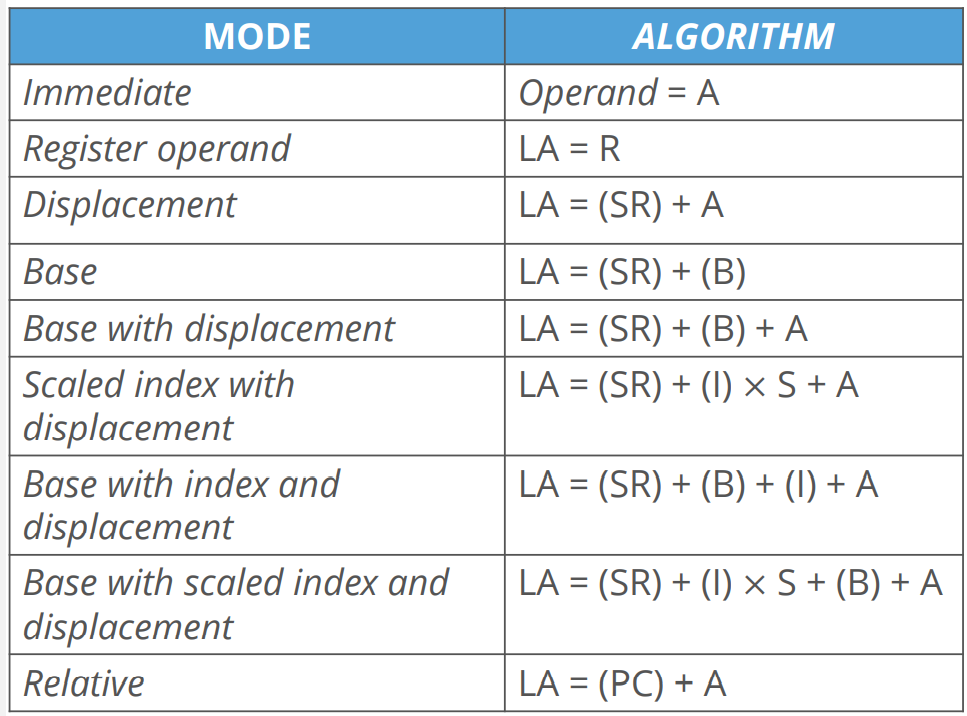
* *Virtual address* merupakan sebuah offset ke sebuah segmen.

Virtual address dilengkapi bermacam-macam *mode addressing* untuk memudahkan bahasa - bahasa pemrograman mengeksekusinya.

Contoh: C atau Fortran.

* *Linear Address* (LA) = alamat awal + offset.

Tabel padaX86 addressing mode

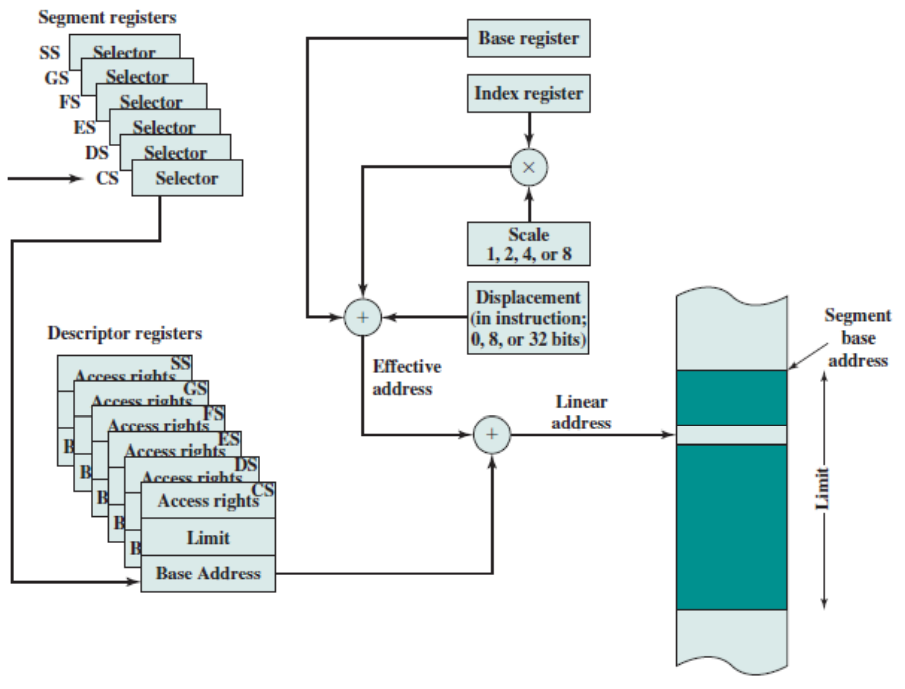


Keterangan :

* LA *= linear address*
* (X) *= contents of X*
* SR *= segment register*
* PC *= program counter*
* A *= contents of an address field in the instruction*
* R *= register*
* B *= base register*
* I *= index register*
* S *= scaling factor*

1. X86 ADDRESSING MODE CALCULATION

Skema di bawah ini adalah X86 addressing *mode calculation*.



Keterangan pada skema :

* *Segmen register* dan *description register* akan masuk melalui linear address, lalu masuk ke segmen base address.
* *Base register* akan masuk melalui effective address, lalu melalui linear address, dan masuk ke segmen base address.
* *Index register dan scale* akan masuk melalui *effective address*, dan masuk ke segmen base address.

1. **X86 ADDRESSING MODE: IMMEDIATE**

Mode immediate pada mode addressing X86:

* Operand berada di dalam instruksi.
* Operand dapat berupa data byte, word maupun double word.

*Immediate Mode* operand register dan operand merupakan sebuah isi register.

Berikut adalah macam-macam register:

* Register 8 bit (AH, BH, CH, DH, AL, BL, CL, DL);
* Register 16 bit (AX, BX, CX, DX, SI, DI, SP, BP);
* Register 32 bit (EAX, EBX, ECX, EDX, ESI, EDI, ESP, EBP);
* Register 64 bit yang dibentuk dari register 32 bit secara berpasangan;
* Register 8, 16, dan 32 merupakan register untuk penggunaan umum;
* Register 64 bit biasanya untuk operasi floating point;
* Register segmen (CS, DS, ES, SS, FS, GS).

1. X86 ADDRESSING MODE: DISPLACEMENT

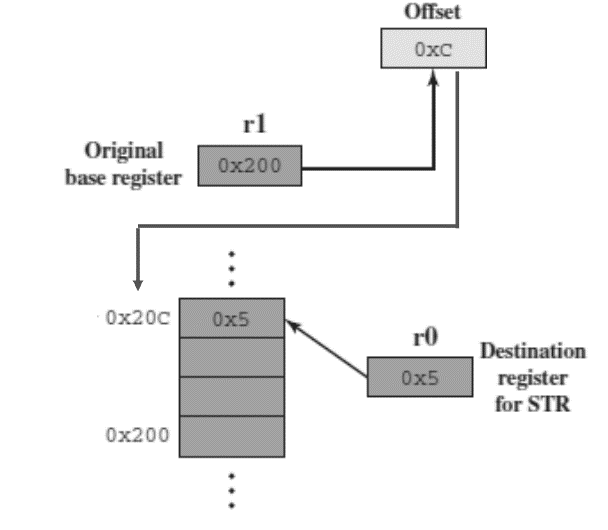
Mode displacement pada *mode addressing* X86: Alamat efektif berisi bagian–bagian instruksi dengan displacement 8, 16, atau 32 bit. Dengan segmentasi, seluruh alamat dalam instruksi mengacu ke sebuah offset di dalam segmen. Pada Pentium, mode ini digunakan untuk mereferensi variabel–variabel global.

1. ARM ADDRESSING MODE

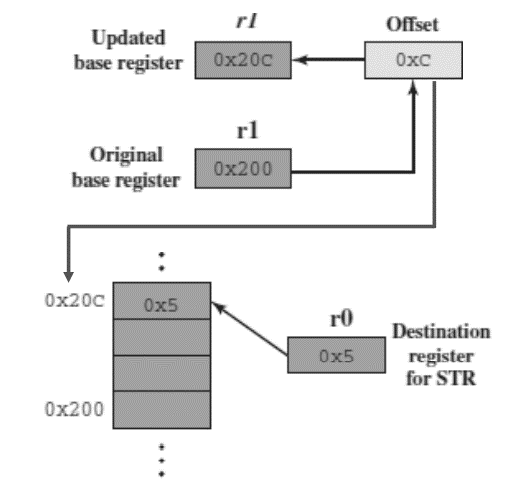
Tidak seperti mesin CISC, mesin RISC menggunakan cara yang sederhana terutama dari mode addressing. Arsitektur ARM sedikit berbeda dengan menyediakan set mode addressing yang relatif banyak. Ini mode yang paling mudah diklasifikasikan sehubungan dengan jenis instruksi.

Terdapat tiga alternatif pada indeks ARM, yaitu:

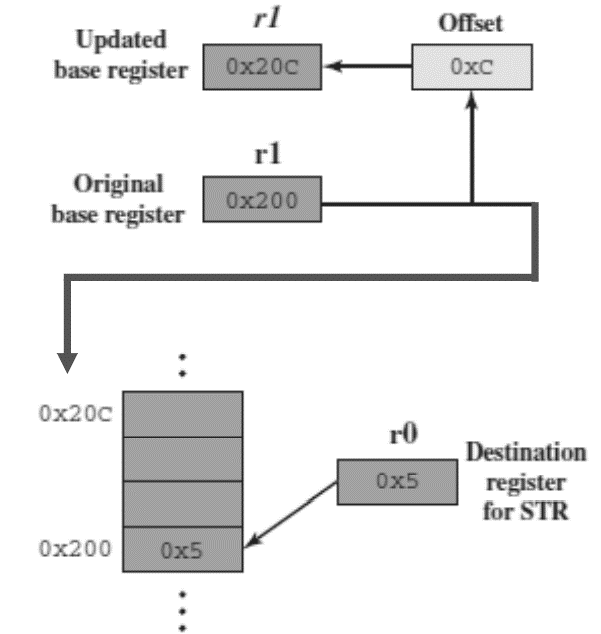
1. Offset *Mode addressing* ini tidak menggunakan indeks. Nilai offset ditambahkan atau dikurangi dari nilai base register untuk membentuk memori address.



1. Preindex Alamat memori dibentuk dengan cara yang sama untuk mengimbangi addressing. Alamat memori juga ditulis kembali ke base register.



1. Postindex Alamat memori merupakan nilai base register. Sebuah offset ditambahkan atau dikurangkan dari nilai base register dan hasilnya ditulis kembali ke base register.



1. PERMASALAHAN DESAIN PENGALAMATAN DAN EFISIENSI PENGALAMATAN

Mode pengalamatan adalah bagaimana cara menunjuk dan mengalamati suatu lokasi memori pada sebuah alamat di mana operand akan diambil.

Terdapat 6 teknik pengalamatan, yaitu:

1. Pengalamatan Register (*Register Addressing*) Metode pengalamatan register mirip dengan mode pengalamatan langsung. Perbedaannya terletak pada *field* alamat yang mengacu pada register, bukan pada memori utama.

Keutungan: Diperlukan field alamat berukuran kecil dalam instruksi dan tidak diperlukan referensi memori, juga akses ke register lebih cepat daripada akses ke memori, sehingga proses eksekusi akan lebih cepat.

Kerugian : Kekurangan pengalamatan register yaitu ruang alamat menjadi terbatas.

1. Pengalamatan Segera (*Immediate Addressing*) : Teknik pengalamatan segera digunakan untuk menentukan dan menggunakan konstanta atau nilai-nilai variabel awal.

Keutungan : Tidak adanya referensi memori selain dari instruksi yang diperlukan untuk memperoleh operand pada saat menghemat siklus instruksi sehingga proses keseluruhan akan lebih cepat

Kerugian : Ukuran bilangan dibatasi oleh ukuran field alamat.

1. Pengalamatan Langsung (*Direct Addressing*) : Teknik pengalamatan langsung merupakan mode pengalamatan yang sederhana yang hanya memerlukan sebuah referensi memori dan tidak memerlukan kalkulasi khusus

Keutungan : Field alamat berisi effective address sebuah operand, teknik ini banyak digunakan pada komputer lama dan mini computer, hanya memerlukan sebuah referensi memori dan tidak memerlukan kalkulasi khusus.

Kerugian : Keterbatasan field alamat karena panjang field alamat biasanya lebih kecil dibandingkan panjang word.

1. Pengalamatan Tak Langsung (*Indirect Addressing*) : Pada mode pengalamatan tak langsung (indirect addressing), field alamat mengacu pada alamat word di dalam memori, yang pada gilirannya akan berisi alamat operand yang panjang :

Keutungan : Ruang bagi alamat menjadi besar sehingga semakin banyak alamat yang dapat dilakukan referensi.

Kerugian : Diperlukan referensi memori ganda dalam satu *fetch* sehingga memperlambat proses operasi.

1. Pengalamatan Register Tak Langsung (*Register Indirect Addressing*) Metode pengalamatan register tidak langsung mirip dengan mode pengalamatan tidak langsung.

Perbedaannya adalah field alamat mengacu pada alamat register

Keutungan: Ruang alamat besar

Kerugian : Referensi memori ekstra

1. *Displacement Addressing* : Mode ini mensyaratkan instruksi memiliki dua

buah field alamat: alamat baru dan alamat lama *yang digantikan*

Keutungan : Fleksibilitas

Kerugian : Kompleksitas

1. PENJAJARAN BATAS (BOUNDARY ALLIGNMENT)Kebanyakan CPU menetapkan alamat memori utama dalam byte, bukan word, sistem penyimpanan biasanya menggunakan alamat word, serta menyimpan dan memanggil kembali word tersebut, akibatnya, sistem penyimpanan atau CPU memerlukan putaran (*circuitry)* khusus untuk operasi byte.
2. **PENGALAMATAN MEMORI**

Semua komputer mempunyai instruksi yang mereferensi memori, umumnya, arsitek mengembangkan teknik pengalamatan ini untuk berbagai macam tujuan. Beberapa arsitek bertujuan agar dukungan hardware efisien dalam mengakses elemen dari array. Beberapa yang lain bertujuan memberikan dukungan untuk menyertakan parameter, sedangkan yang lain bertujuan agar terjadi spesifikasi alamat yang kecil untuk menghasilkan alamat yang besar, umumnya, teknik pengalamatan memberikan dukungan lebih dari satu fungsi.

1. **PERMASALAHAN DESAIN PENGALAMATAN**

Para arsitek harus mempertimbangkan beberapa fasilitas pengalamatan ketika ia merancang komputer atau famili komputer. Tiga fasilitas pengalamatan dalam permasalahan desain pengalamatan :

1. **Jangkauan Pengalamatan** : Pengalamatan fisik adalah suatu metode pengalamatan memori secara fisik. Alamat fisik memori utama yang sesungguhnya merupakan alamat yang ditulis pada kode instruksi program hasil kompilasi. Pada saat penyalinan ke memori utama, kode instruksi dan data program disalin pada posisi yang sesuai dengan referensi tersebut. Ketika eksekusi, prosesor akan memproses alamat kode instruksi program secara langsung tanpa melakukan translasi memori.
2. **Homogenitas Pengalamatan** Ruang alamat homogen adalah salah satu dari fasilitas kemudahan penggunaan set instruksi komputer. Dalam homogeneous addressing (pengalamatan homogen) semua alamat dianggap sama, sedangkan dalam pengalamatan tidak homogen beberapa alamat dianggap melebihi yang lainnya. Jika ruang alamat tidak homogen, maka programmer dan compiler harus mengadaptasi pengalamatan yang mereka hasilkan dengan ketidakhomogenan yang ada di dalam ruang alamat.
3. **Efisiensi Pengalamatan** Efisiensi pengalamatan adalah sejauh mana efektivitas programmer atau compiler dapat menggunakan mode pengalamatan yang telah ditetapkan oleh ISA. Arsitektur RISC dilengkapi dengan beberapa mode pengalamatan yang sederhana, sedangkan arsitektur CISC dilengkapi dengan mode pengalamatan sederhana yang banyak, biasanya berupa mode pengalamatan yang kompleks.

**REFERENSI/DAFTAR PUSTAKA**

1. Stallings, W. (2013). Computer organization and architecture (9th ed.). UK: Pearson.