# Tugas II

Nama : Nur Hasna’ Shofia

NIM : L200150056

Kelas : A

# Set instruksi

* **Set Instruksi** (bahasa [Inggris](https://id.wikipedia.org/wiki/Inggris): **Instruction Set**, atau **Instruction Set Architecture** (ISA)) didefinisikan sebagai suatu aspek dalam [arsitektur komputer](https://id.wikipedia.org/wiki/Arsitektur_komputer) yang dapat dilihat oleh para pemrogram. Secara umum, ISA ini mencakup jenis [data](https://id.wikipedia.org/wiki/Data) yang didukung, jenis [instruksi](https://id.wikipedia.org/w/index.php?title=Instruksi&action=edit&redlink=1) yang dipakai, jenis [register](https://id.wikipedia.org/wiki/Register), [mode pengalamatan](https://id.wikipedia.org/w/index.php?title=Mode_pengalamatan&action=edit&redlink=1), [arsitektur memori](https://id.wikipedia.org/w/index.php?title=Arsitektur_memori&action=edit&redlink=1), penanganan [interupsi](https://id.wikipedia.org/wiki/Interupsi), [eksepsi](https://id.wikipedia.org/w/index.php?title=Exception&action=edit&redlink=1), dan operasi I/O eksternalnya (jika ada).
* ISA merupakan sebuah spesifikasi dari Pullman semua kode-kode [biner](https://id.wikipedia.org/wiki/Biner) (*opcode*) yang diimplementasikan dalam bentuk aslinya (*native form*) dalam sebuah desain [prosesor](https://id.wikipedia.org/wiki/Mikroprosesor) 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](https://id.wikipedia.org/wiki/X86), [IA-64](https://id.wikipedia.org/w/index.php?title=IA-64&action=edit&redlink=1), [IBM PowerPC](https://id.wikipedia.org/w/index.php?title=IBM_PowerPC&action=edit&redlink=1), [Motorola 68000](https://id.wikipedia.org/w/index.php?title=Motorola_68000&action=edit&redlink=1), [Sun SPARC](https://id.wikipedia.org/w/index.php?title=Sun_SPARC&action=edit&redlink=1), [DEC Alpha](https://id.wikipedia.org/wiki/Alpha_AXP), 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*](https://id.wikipedia.org/w/index.php?title=Microcode&action=edit&redlink=1), [*pipeline*](https://id.wikipedia.org/wiki/Pipeline), [*sistem cache*](https://id.wikipedia.org/w/index.php?title=Sistem_cache&action=edit&redlink=1), [manajemen daya](https://id.wikipedia.org/w/index.php?title=Manajemen_daya&action=edit&redlink=1), dan lainnya). Komputer-komputer dengan [mikroarsitektur](https://id.wikipedia.org/w/index.php?title=Mikroarsitektur&action=edit&redlink=1) berbeda dapat saling berbagi set instruksi yang sama. Sebagai contoh, prosesor [Intel Pentium](https://id.wikipedia.org/wiki/Intel_Pentium) dan prosesor [AMD Athlon](https://id.wikipedia.org/wiki/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](https://id.wikipedia.org/w/index.php?title=IBM_System/38&action=edit&redlink=1) dan [IBM IAS/400](https://id.wikipedia.org/w/index.php?title=IBM_IAS/400&action=edit&redlink=1). 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](https://id.wikipedia.org/w/index.php?title=AS/400&action=edit&redlink=1) dari arsitektur mikroprosesor [CISC](https://id.wikipedia.org/wiki/CISC) ke arsitektur mikroprosesor [POWER](https://id.wikipedia.org/w/index.php?title=POWER&action=edit&redlink=1) 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](https://id.wikipedia.org/w/index.php?title=Register_Transfer_Language&action=edit&redlink=1) (RTL) untuk mendefinisikan operasi dari setiap instruksi yang terdapat dalam ISA.
* Sebuah ISA juga dapat diemulasikan dalam bentuk [perangkat lunak](https://id.wikipedia.org/wiki/Perangkat_lunak) oleh sebuah [interpreter](https://id.wikipedia.org/wiki/Interpreter). Karena terjadi translasi tambahan yang dibutuhkan untuk melakukan [emulasi](https://id.wikipedia.org/w/index.php?title=Emulasi&action=edit&redlink=1), 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](https://id.wikipedia.org/wiki/Perangkat_keras) dirilis oleh vendor.

# [ARSITEKTUR SET INSTRUKSI](http://irfan-abet.blogspot.com/2015/01/arsitektur-set-instruksi.html)

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.

**ELEMEN - ELEMEN DARI SET INSTRUKSI**

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

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

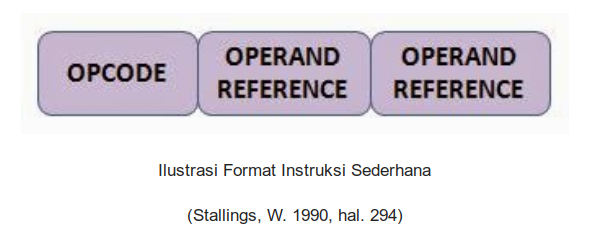
c. *Result Operand Reference* : merupakan hasil dari operasi yang dilaksanakan.

d. *Next Instruction Reference* : memberitahu CPU untuk mengambil instruksi berikutnya setelah instruksi yang dijalankan selesai.

**FORMAT INSTRUKSI**

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

Contoh suatu Format Instruksi adalah sbb :



**JENIS - JENIS INSTRUKSI**

1. *Data Processing* / Pengolahan Data : instruksi-instruksi aritmetika dan logika. Instruksi aritmetika memiliki kemampuan untuk mengolahdata numeric, sedangkan instruksi logika beroperasi pada bit-bit word sebagai bit bukan sebagai bilangan. Operasi-operasi tersebut dilakukan terutama untuk data di register CPU.

2. *Data Storage* / Penyimpanan Data : instruksi-instruksi memori. Instruksi-instruksi memori diperlukan untuk memindah data yang terdapat di memori dan register.

3. *Data Movement* / Perpindahan Data : instruksi I/O. Instruksi-instruksi I/O diperlukan untuk memindahkan program dan data ke dalam memori dan mengembalikan hasil komputansi kepada pengguna.

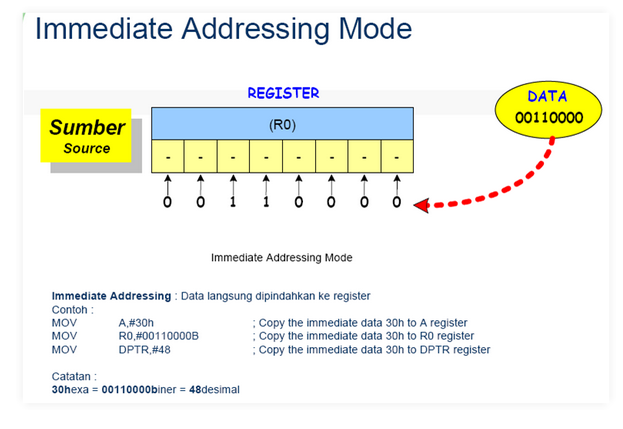
4. *Control*/ Kontrol : instruksi pemeriksaan dan percabangan. Instruksi-instruksi kontrol digunakan untuk memeriksa nilai data, status komputansi dan mencabangkan ke set instruksi lain.

**TEKNIK PENGALAMATAN**

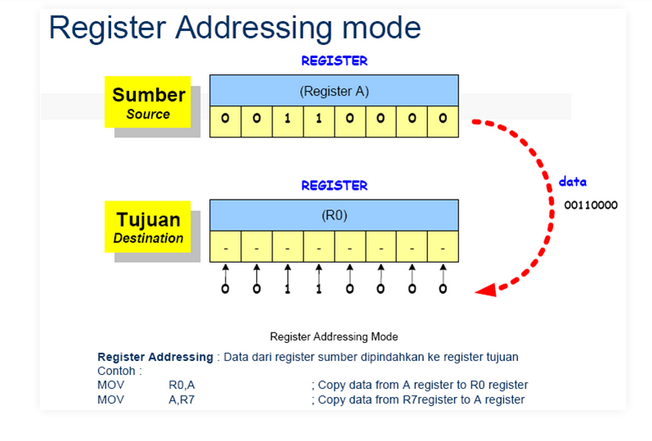
Metode pengalamatan merupakan aspek dari set instruksi arsitektur disebagian unit pengolah pusat (CPU) desain yang didefinisikan dalam set instruksi arsitektur dan menentukan bagaimana bahasa mesin petunjuk dalam arsitektur untuk mengidentifikasi operan dari setiap instruksi. Sebuah mode pengalamatan menentukan bagaimana menghitung alamat memori yang efektif dari operand dengan menggunakan informasi yang diadakan di register dan / atau konstanta yang terkandung dalam instruksi mesin atau di tempat lain.

Jenis-jenis metode pengalamatan diantaranya :

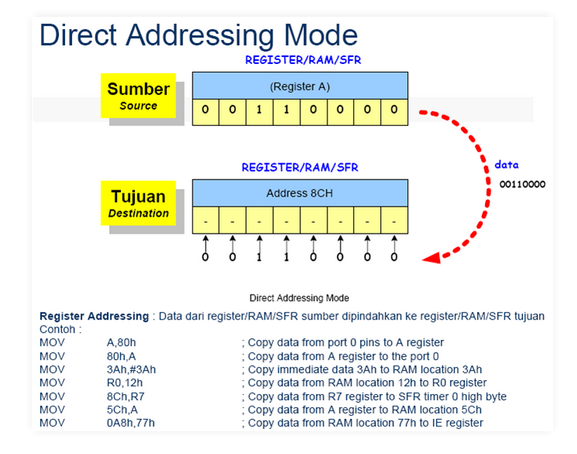
1. *Immediate Addressing Mode*



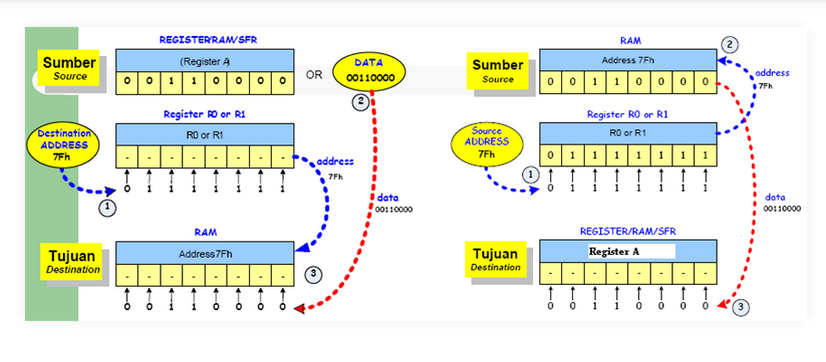
2. *Register Addressing Mode*



3.*Direct Addressing Mode*



4. *Indirect Addressing Mode*



R0 atau R1 digunakan untuk menunjukkan *Destination Address*

MOV A,#30h   ; salin immediate data 30h ke Akumulator

MOV R0,#7Fh   ; salin immediate data 7Fh ke register R0

MOV @R0,A   ; salin the data in A ke alamat di R0

R0 atau R1 digunakan untuk menunjukkan *Source Address*

MOV R0,#7Fh   ; salin immediate data 7Fh ke register R0

MOV @R0,#30h  ; salin immediate data 30 ke alamat di R0

MOV A,@R0   ; salin isi dari alamat di R0 ke Akumulator

**DESAIN SET INSTRUKSI**

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

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 :

a. *Operation Repertoire*, berapa banyak dan operasi apa saja yang disediakan dan berapa sulit operasinya.

b. *Data Types*, tipe / jenis data yang dapat diolah.

c. *Instruction Format*, panjangnya, banyaknya alamat, dsb.

d. *Register*, banyaknya register yang dapat digunakan.

e. *Addressing*, mode pengalamatan untuk operand.

### SET INTRUKSI PADA MIKROPROSESOR 8086

***Instructions Set pada microprosesor 8086***

***1. Instructions Set***

Adalah sekumpulan instruksi lengkap yang dapat dimengerti oleh CPU, instruction sets  berupa kode mesin (machine code) dalam bentuk bilangan biner (binary) dan biasanya di- representasi-kan dalam kode/bahasa rakitan (assembly code).

***a. Element dari instruction***

Instruction terdiri dari beberapa element, yaitu:

ØOperation code (Op code)

·Kode perintah operasi .

ØSource Operand reference

·Operand penampung nilai yang akan dipros

Ø Result Operand reference

·Operand penampung nilai hasil proses.

ØNext Instruction Reference

·Penghubung ke instruksi berikutnya.

Operand merupakan obyek dari suatu Op code, operand biasanya ditampung pada salah satu tempat penyimpanan berikut:

ØMain memory (or virtual memory or cache)

ØCPU register

ØI/O device

***b. Representasi dari Op code***

ØRepresentasi dalam bahasa/kode mesin (machine code) tiap instruction memiliki  bentuk/pola bit yang unik (unique bit pattern).

ØRepresentasi symbolic atau kode/bahasa rakitan (assembly code), lebih manusiawi  well, programmers anyway), contohnya:

·ADD

·SUB

·LOAD

Ø Operands can also be represented in this way

·ADD A,B

***c. Tipe – tipe Instruction***

Ø Data processing

Ø Data storage (main memory)

Ø Data movement (I/O)

ØProgram flow control

***d.Tipe – tipe Operand***

Øadresses

Ø Numbers

·nteger/floating point

ØCharacters

·ASCII etc.

ØLogical Data

·Bits or flags

***e. Tipe – tipe Data pada Prosessor Pentium***

Tipe – tipe data yang dapat ditampung pada prosessor pentium adalah tipe data dengan besar kapasitas tampung sebagai berikut:

Ø8 bit (Byte)

Ø16 bit (word)

Ø32 bit (double word)

Ø64 bit (quad word)

Keterangan:

Tiap data dialamatkan (Addressing) per- 8 bit untuk tiap unit dengan demikian  untuk yang tipe 32 bit (double word) akan di bagi menjadi 4 unit

***f. Tipe – tipe Operasi***

Berikut adalah tipe – tipe operasi yang terkait dengan Op code:

Ø Operasi Data Transfer

Operasi tipe ini melakukan transfer data, pada operasi tranfer data perlu di tentukan:

· Source = sumber data

· Destination = tujuan operasi

· Amount of Data = besaran/nilai data

Ø Operasi Arithmetic

Operasi tipe ini melakukan operasi – operasi aritmatika sebagai berikut:

· Add

· Subtract

· Multiply

· Divide

Ø Operasi Logical

Operasi tipe ini melakukan operasi yang bersifat logical sebagai berikut:

· ƒ AND

· ƒ OR

· ƒ NOT

Ø Operasi Conversion

Operasi tipe ini melakukan operasi konversi bilangan, seperti:

· ƒ Konversi bilangan biner ke desimal

· ƒ Konversi bilangan real ke int (bilangan bulat)

Ø Operasi I/O

Operasi tipe ini merupakan operasi yang berhubungan dengan Input/Output,

contohnya:

· ƒ Operasi untuk menampilkan suatu nilai ke layar monitor

· ƒ Operasi untuk mencetak, dsb

Ø Operasi System Control

Operasi tipe ini biasanya operasi untuk penggunaan sistem operasi

Ø Operasi Transfer of Control

Operasi tipe ini merupakasn operasi yang mengendalikan flow dari proses,

biasanya berupa percabangan, contohnya:

· ƒ branch to x if result is zero

· ƒISZ Register1 Æ ISZ artinya: Increment and Skip if Zero

· ƒ interrupt call

***g. Mode Pengalamatan (Addressing Mode) untuk Operand***

Terdapat 7 mode pengalamatan operand, yaitu:

Ø Immediate

Ø Direct

Ø Indirect

Ø Register

Ø Register Indirect

Ø Displacement (Indexed)

Ø Stack

***h. Mode Immediate Addressing***

Merupakan mode pengalamatan operand secara langsung, pada mode ini operand merupakan bagian dari instruction. Operand merupakan area alamat (address field) dari suatu nilai yang akan diproses

Contoh: ADD 5

Keterangan: Tambahkan nilai 5 dengan nilai di register accumulator dan simpan hasilnya di register accumulator.

***Karakteristik mode immediate:***

· No memory reference to fetch data Æ tidak memakai referensi memory untuk mengambil data

· Fast Æ cepat

· Limited range Æ terbatas dalam jangkauan nilai

Diagram untuk mode immediate addressing:

**1. Mode Direct Addressing**

Merupakan mode pengalamatan operand dimana area alamat (address field) berisi alamat dari suatu nilai yang akan diproses.

Effective Address (EA) = Address field (A)

EA = A

Contoh: ADD A

*Keterangan:*

· Cari di memory pada alamat A untuk operand (*Look in memory at address A for operand*).

· Tambahkan isi yang ada pada alamat A dengan nilai di register accumulator dan simpan hasilnya di register accumulator. (*Add contents of cell A to accumulator*).

*Karakteristik:*

· *Single memory reference to access data* ?Menggunakan memory untuk mengakses data

· *No additional calculations to work out effective address* ?Tidak memerlukan kalkulasi untuk mendapatkan effective address

· *Limited address space* ?Address space yang terbatas.

Diagram untuk mode direct addressing:

**2. Mode Indirect Addressing**

Merupakan mode pengalamatan operand dimana area alamat (address field) berisi alamat dari suatu alamat yang akan menunjukkan alamat dari suatu nilai yang akan diproses. (*Memory cell pointed to by address field contains the address of (pointer to) the operand*)

EA = ( A )

Keterangan:

· Cari di memory alamat A, cari alamat yang tertulis pada A untuk operand ( *Look in A, find address (A) and look there for operand* ).

Contoh: ADD ( A )

*Keterangan:*

· Tambahkan isi dari cell yang alamatnya ditunjukkan oleh isi yang terdapat pada A dengan nilai yang ada di register accumulator dan simpan hasilnya di register accumulator. (*Add contents of cell pointed to by contents of A to accumulator*).

*Karakteristik:*

· *Large address space*» memerlukan space address yang besar.

· *May be nested, multilevel, cascaded*» Dapat dibuat nested (bersarang), multilevel dan cascade (bertumpuk).

· *Multiple memory accesses to find operand* ?pengaksesan memory yang multiple untuk mendapatkan operand sehingga mengakibatkan proses mode ini agak lebih lambat

Diagram untuk mode direct addressing:

**3. Mode Register Addressing**

Merupakan mode pengalamatan operand dimana operand yang akan diproses ditampung/disimpan dalam register yang namanya ditulis di area alamat (address field).  (*Operand is held in register named in address field*)

EA = R

*Karakteristik:*

· *Limited number of registers*» terbatas pada jumlah register yang hanya sedikit

· *Very small address field needed* »  karena addres yang kecil pada register, maka:

o *Shorter instructions* » instruksinya lebih pendek

o *Faster instruction fetch* » pemasukan data lebih cepat

o *No memory access* » tidak memerlukan akses ke memory

o *Very fast execution* » eksekusi sangat cepat

o *Very limited address space* » tetapi space alamat (*address*) sangat terbatas

Diagram untuk mode direct addressing:

**4. Mode Register** **Indirect Addressing**

Merupakan mode pengalamatan operand dimana area alamat (address field) pada register berisi alamat dari suatu alamat yang akan menunjukkan alamat di memory dari suatu nilai yang akan diproses.  ( *Operand is in memory cell pointed to by contents of register R* )

EA = ( R )

Diagram untuk mode direct addressing:

**Displacement** **Addressing**

**EA = A + (R)**

Berdasarkan formula di atas, address field menampung 2 nilai, yaitu:

Ø A sebagai base value

Ø R sebagai register yang menampung pertukaran sementara (*that holds displacement*)

Ø Atau sebaliknya

***Relative*** ***Addressing***

Merupakan salah satu versi dari pengalamatan untuk pertukaran ( *displacement addressing* )

R = Program counter, PC

EA = A + (PC)

*Keterangan:*

· *Ambil* *operand* dari A dan dari lokasi yang ditunjukkan oleh program counter (PC) (*get operand from A cells from current location pointed to by PC*)

**Micro Operation**

· *Suatu komputer menjalankan suatu program (A computer executes a program)*

· Suatu komputer melakukan siklus proses memasukkan dan mengambil data atau melakukan eksekusi (*Fetch/execute cycle*)

· *Each* cycle has a number of steps

· Aktifitas tersebut di atas disebut micro-operations

· Tiap step pada micro-operations berupa langkah – langkah kecil (*Each step does very little*).

· Micro-operations merupakan operasi atomic dari CPU (*Atomic operation of CPU*)

***Fetch***

***4 Registers***

· **Memory Address Register** (**MAR**)

o Terkoneksi dengan address bus

o MAR melakukan spesifikasi address untuk operasi baca atau tulis

· *Memory* **Buffer Register** (**MBR**)

o Terkoneksi dengan data bus

o Menyimpan data untuk di tulis atau menyimpan data terakhir yang dibaca (*Holds data to write or* last *data read*)

· **Program Counter** (**PC**)

o Menyimpan address instruksi berikutnya yang akan diakses (*Holds address of next instruction to be fetched*)

· **Instruction Register** (**IR**)

o Menyimpan address instruksi terakhir yang diakses (*Holds last instruction fetched*)

***Fetch* Sequence** ***(Urutan Fetch)***

· Address dari instruksi berikutnya berada dalam PC

· Address yang ada pada MAR di masukkan ke address bus

· Control Unit memerintahkan perintah membaca (*Control unit issues READ command*)

· Hasil (data dari memory) berada pada data bus

· Data dari data bus di copy ke MBR

· PC ditambah 1 (secara paralel dengan proses fetch dari memory)

· Data (instruksi) dipindahkan dari MBR ke IR

· MBR sekarang kosong dan siap untuk fetch selanjutnya.

***Fetch Sequence Symbolic (Urutan Fetch secara simbol)***

t1: MAR » (PC)

t2: MBR » (memory)

PC» (PC) +1

t3: IR » (MBR)

(tx = time unit/clock cycle)

Or

t1: MAR » (PC)

t2: MBR » (memory)

t3: PC » (PC) +1

IR » (MBR)

***Aturan untuk pengelompokan siklus waktu (Rules for Clock Cycle Grouping)***

· Urutan yang baik harus mengikuti aturan berikut (*Proper sequence must be followed*) :

o MAR » (PC) harus diikuti dengan MBR » (memory)

·Harus menghindari terjadinya konflik (*Conflicts must be avoided*)

o Tidak boleh melakukan pembacaan dan penulisan terhadap register yang sama dan pada waktu yg bersamaan (*Must not read & write same register at same time*)

o Aktifitas MBR » (memory) & aktifitas IR » (MBR) tidak boleh dilakukan pada siklus yang bersamaan

·PC harus ditambahkan 1: PC » (PC) +1

o Menggunakan ALU

o Mungkin diperlukan additional micro-operations

***Siklus tidak Langsung (Indirect Cycle)***

· MAR » (IRaddress) artinya: Alamat dari IR diisikan ke MAR

· MBR »  (memory) artinya: isi dari memory diisikan ke MBR

· IRaddress »  (MBRaddress) artinya: alamat dari MBR diisikan ke alamat dari IR

· MBR berisi suatu address (*MBR contains an address*)

· Sekarang IR berada pada kondisi yang sama dengan direct addressing (*IR is now in same state as if direct addressing had been used*)

### set instruksi mikroprosesor 8088

**Arsitektur Microprocessor 8088**

Di awali dari definisi Mikroprosesor itu sendiri, ialah sebuah chip (IC) yang bekerja dengan program. Fungsinya adalah sebagai pengontrol atau pengolah utama dalam suatu rangkaian elektronik. [Mikroprosesor](http://de-kill.blogspot.com/2009/05/sekilas-sejarah-mikroprosesor.html) biasa disebut juga dengan CPU (Central Processing Unit). Sekilas tentang sejarahnya yaitu pada tahun 1978  Intel mengeluarkan mikroprosesor 16-bit yaitu 8086, setahun kemudian mengeluarkan 8088 dengan kecepatan eksekusi dan memori lebih besar dari 8085, serta mulai digunakannya cache memori (sistem antrian yang mengatur pemberian instruksi sebelum menjalankannya.

Mikroprosesor Intel 8088 termasuk keluarga mikroprosesor 8 bit dan 16 bit. Mikroprosesor 8088 mempunyai 8 bit jalur data dan 20 bit jalur alamat. Jalur data memiliki pin yang sama dengan jalur alamat, artinya pada saat tertentu digunakan sebagai jalur data dan pada saat yang lain digunakan sebagai jalur alamat.

Setiap satu pin memiliki dua fungsi, yaitu sebagai jalur data dan jalur alamat maka digunakanlah sistem time multiplexing, yaitu penggunaan jalur yang sama untuk fungsi dan waktu yang berbeda, sehingga tidak bisa data dan alamat dikirim pada saat yang bersamaan.

Mikroprosesor 8088 mampu mengalamati memori sampai 1 MB. Memori ini digunakan untuk menyimpan kode biner dari instruksi yang akan dijalankan oleh μP, selain itu memori juga digunakan untuk menyimpan data secara sementara.

**Set Instruksi Pada Microprocessor 8088**

1. Transfer Data

Set instruksi 8088 mencakup 14 transfer data yang menggerakkan byte atau kata data antara memori dan register selain juga antara akumulator dan port I/O.

a. General Purpose (Umum) : MOV, PUSH, POP, XCHG, SXLAT

b. Input/Output: In, Out

c. Address Transfer (Pemindahan Alamat) : LEA, KDS, KES

d. Flag Transfer (Pemindahan Flag) : LAHF, SAHF, PUSHF, POPF

2. Arithmetic Instructions (Instruksi Perhitungan)

8088 mampu melakukan penambahan, pengurangan, pengurangan dan pembagian data selain juga bytes dan kata. Sistem ini menambah dan mengurangi dengan menggunakan bytes atau kata yang bertanda atau tak bertanda dan data BCD atau ASCII.

a. Addition (Penjumlahan) : ADD, ADC, AAA, DAA, INC

b. Substraction (Pengurangan) : SUB, SBB, AAS, DAS, DES, NEG, CMP

c. Multiplication (Perkalian) : MUL, IMUL, AAM

d. Division (Pembagian) : DIV, IDIV, AAD

e. Sign Extension : CBW, CWD

3. Manipulasi Bit

Instruksi ini mencakup operasi logika, shift dan rotasi.

a. Logical (Logika) : AND, OR, XOR, NOT, TEST

b. Shift (Geser) : SAL/SHL, SAR, SHR

c. Rotate (Gulung) : ROL, ROR, RCL, RCR

4. Instruksi String

Instruksi string digunakan untuk memanipulasi string data dalam memori. Setiap string tersusun baik dari bytes maupun kata dan hingga mencapai 64 K bytes panjang.nya

a. Direction: CLD, STD

b. Repeat Prefixes : REP, REPE/REPZ, REPNE/REPNZ,

c. Move : MOVSB, MOVSW

d. Compare: CMPS, CMPSB, CMPSW

e. Scan: CSAS, SCASB, SCASW

d. Load and Store : LODS, LODSB, LODSW, STOS, STOSB, STOSW

5. Transfer Program

Instruksi transfer program mencakup *jump,  call*, dan *return*, instruksi yang sudah kita kenal dalam 8085.

a. Unconditional transfer (Pemindahan) : CAL, RET, JMP

b. Conditional transfer (Pemindahan Bersyarat) : JA/JNBE, JAE/JNB, JB/JNAE/JC, JBE/JNA, JCXZ, JE/JZ, JG/JNLE, JGF/JNL, JL/JNGE, JLE/JNG, JNC, JNE/JNZ, JNO, JNP/JPO, JNS, JO, JP/JPE, JS

c. Loop/Jump: LOOP, LOOPE/LOOPZ, LOOPNE/LOOPNZ

6. Kontrol Prosesor

Instruksi kontrol prosesor memungkinkan dan tidak memungkinkan interupsi, memodifikasi bit flag dan mensinkronisasi kejadian eksternal.

a. Flag Operations : CLC, STC, CMC, CLD, STD, CLI, STI,

b. External Synchronization

7. Interrupt : INT, INTO, IRET

SEKILAS INTEL 8088

Intel 8088 adalah prosesor mikro buatan Intel berbasis pada 8086, dengan 16-bit register dan menggunakan 8-bit external data bus. intel 8088 merupakan prosesor yang digunakan pada IBM PC.

8088 ditargetkan pada sistem yang ekonomis, diikuti oleh pengunaan desain 8-bit. Jalur bus yang lebar dalam circuit boards masih sangatlah mahal ketika ini di luncurkan. Queue yang ungul dari 8088 adalah 4 bytes, sebagai penggunaan dalam 8086 6 bytes. 8088 termasuk keturunan dari 80188, 80288, 80186, 80286, 80386, 80486, dan 80388, microcontroller seperti yang masih digunakan sekarang. Clone yang populer dengan menggunakan 8088 adalah Model D, dimana tombol pilihan dapat berjalan pada clock 4.77 MHZ atau 7.16 MHZ.

Spesifikasi 8088

Mikroprosesor Intel 8088 hampir serupa dengan prosesor Intel 8086, kecuali pada data  eksternal bus. Lebar data eksternal bus 8088 dikurangi menjadi 8 bit, dan instruksi ukuran queue dan prefetching algoritmanya diubah. Intel 8088 menggunakan dua urutan bus siklus untuk menulis atau membaca 16 data bit sebagai ganti satu siklus untuk 8086. Ini menjadikan prosesor  bergerak  lebih lambat, tetapi ada nilai plus pada perangkat keras yang menjadikan CPU 8088 kompatibel dengan peripheral 8080/8085.

Pin SSO pada 8088 menggantikan BHE/S7 pada 8086, dan pin IO/M pada 8088, bukan M/IO seperti pada 8086. 8088 membutuhkan catu daya +5,0 V dengan toleransi + 10%. Mikroprosesor 8088 akan kompatibel TTL (Transistor-Transistor Logic) jika kekebalan terhadap noise disesuaikan menjadi 350 mV dari nilai 400 mV yang biasa. Mikroprosesor 8088 dapat menjalankan satu 74XX, lima 74LSXX, satu 74SXX, sepuluh 74ALSXX, dan sepuluh 74HCXX beban satuan. Jika mikroprosesor 8088 direset, mikroprosesor ini mulai mengeksekusi perangkat lunak pada lokasi meori FFFF0H (FFF:0000) dengan pin interrupt request di-disable. Karena bus-bus 8088 dimultipleks dan kebanyakan memori dan peralatan I/O tidak, system harus didemultipleks sebelum pengantarmukaan dengan memori atau I/O. Demultipleks dilakukan oleh latch delapan bit yang pulsa clocknya didapat dari sinyal ALE. Operasi minimum 8088 sama dengan mikroprosesor Intel 8085A, sementara mode maksimum adalah baru dan khusus dirancang untuk operasi koprosesor aritmatika 8087.

REGISTER

Sebuah register adalah sebuah tempat penampungan sementara untuk data-data yng akan diolah oleh prosesor, dan dibentuk oleh 16 titik elektronis di dalam chip mikroprosesor itu sendiri. Dengan adanya tempat-tempat penampungan data sementara ini, proses pengolahan akan bisa dilakukan

secara jauh lebih cepat dibandingkan apabila data-data tersebut harus diambil langsung dari lokasi-lokasi memori. Register-register tersebut sebagai register internal dan terdiri dari empat belas register dan keseluruhannya dapat dibagi dalam beberapa jenis, yaitu : Register Segment, Register Data, Register pointer, Register index, Register index, dan General Purpose Register.

Semua general register mikroprosesor  8088 dapat digunakan untuk perhitungan dan operasi logika.

Pada 8088, register data diwujudkan oleh AX, BX, CX dan BX (sebagai general purpose register), sehubungan dengan fungsinya yang selain menangani tugas-tugas khusus, juga bisa dimanfaatkan untuk membantu proses-proses pengolahan data didalam internal mikroprosesor.

Mikroprosesor 8088 mempunyai kemampuan untuk bekerja dalam mode 'langkah tunggal' (single-step), yaitu semua instruksi dilaksanakan dengan cara satu demi satu. Mode ini dimungkinkan

dengan jalan membuat TF (Trap Flag) masuk ke logika '1' atau 'set'. Bagi seorang programmer, mode ini akan sangat berguna dalam pekerjaan 'debugging'.

Organisasi Ruang Memori Dan Register

Unsur terkecil memori adalah 'sel memori' (memory-cell), yaitu suatu elemen penyimpanan data yang berkapasitas sebesar 1 bit. Dengan menggabungkan sejumlah sel memori, akan bisa membentuk suatu ruang penyimpanan data dengan berbagai ukuran, misalnya 1 byte, 1 word, 1 Kilobyte, 1 Megabyte, 1 Gigabyte, 1 Terabyte, dsb.

Organisasi memori dapat dibandingkan dengan sistem 'locker' (susunan laci yang mempunyai kode nomor setiap lacinya sehingga memudahkan orang mengenal lacinya masing-masing sebelum mengambil atau memasukkan barang titipannya).

Susunan chip prosesor Intel 8088.

• Catu Daya/VCC (pin 40) dan GND (pin 1 dan 20)

• Bus Data (AD0 - AD7)

• Bus Alamat (AD0 - AD7 dan A8 - A19)

• Bus Kendali (NMI, INTR, CLK, Reset).

Dalam mikroprosesor 8088 secara fisik, bus alamat terdiri dari 20 bit (A0-A19), sementara register-register internal terbentuk dari 16 bit data. Oleh sebab itu, untuk menyesuaikan perbedaan jumlah bit antara bus alamat 8088 dengan register internal, sistem pengalamatan memori dilaksanakan

dengan format segment:offset. Format yang membutuhkan 32 bit ini dibentuk dengan jalan menggabungkan data dari 2 buah register sekaligus. Register pertama adalah satu satu dari 4 register segment, sedangkan register lainnya diambil dari salah sebuah register pointer atau register indeks.

Kenyataannya, segment-segment yang didefinisikan pada ruang memori itu boleh dibuat saling berdampingan, terpisah atau tumpang tindih sekalipun. Prosesor memiliki bus alamat sebanyak 20 bit, yang berarti ia mampu mengalamati hingga 1.048.575 lokasi memori. Secara heksadesimal, jumlah ini dinyatakan sebagai angka 00000 sampai  dengan FFFFF.  Ini adalah alamat-alamat fisik (physical addresses) dari mikroprosesor. Untuk 8088 dan 8086 yang bus alamatnya terdiri dari 20 bit, otomatis penulisan alamat fisiknya terdiri dari 5 digit heksadesimal. Sistem segmentasi pada IBM PC dilaksanakan agak unik. 1 segment adalah bagian dari ruang memori yang besarnya 65536 byte atau 64 Kb. Namun, segment-segment itu tidaklah diletakkan secara berdampingan sambung menyambung satu sama lain, akan tetapi saling tumpang tindih sehingga jarak antara titik awal suatu segment hanya terpaut 16 byte terhadap segment lainnya.

Peta Memori (Memory Map)

Kapasitas memori untuk IBM PC/XT yang berbasis prosesor Intel 8088/8086 adalah 1.048.576 byte atau lebih mudah disebut 1 (satu) Megabyte. Nilai sebesar 1 MB inilah yang menjadi dasar sistem pemetaan memori dalam keluarga IBM PC Kompatibel, sehingga dalam produk-produk yang lebih mutakhir pun, peta memori tersebut tetap dipertahankan. Hal ini berhubungan dengan konsistensi yang harus dijaga pada Disk Operating System, yang dalam keadaan bagaimanapun, harus tetap bisa dijalankan mulai dari produk yang paling awal seperti PC/XT, sampai kepada yang terbaru seperti AT 486 kompatibel.

ARSITEKTUR

Arsitektur dari 8088 tetap sama degan 8086 yakni: 16-bit registers, 16-bit internal data bus dan 20-bit address bus, yang bisa menjadikan prosesor mencapai memori 1 MB. 8088 memiliki pembagian memori yang sama dengan 8086: prosesor bisa mencapai 64 KB dari memori secara langsung, dan untuk mencapai lebih dari 64 KB, salah satu dari bagian khusus register harus di update.

Program, data dan stack memori menduduki ruang memori yang sama. Total kapasitas memory yang bisa dicapai adalah 1MB KB. Sebagaimana kebanyakan instruksi prosesor yang menggunakan 16-bit pointers, prosesor dapat mengolah secara efektif jika hanya memorinya 64 KB. Untuk mengakses memori diluar 64 KB,  CPU menggunakan bagian  register khusus untuk menspesifikasi di mana kode, stack dan 64 KB segmen data diposisikan di dalam memori 1 MB.

16-bit pointers dan data disimpan sebagai:

address: low-order byte

address+1: high-order byte

32-bit addresses disimpan di "segment:offset" dengan format:

address: low-order byte of segment

address+1: high-order byte of segment

address+2: low-order byte of offset

address+3: high-order byte of offset

Physical memory address ditunjukkan oleh pasangan segment:offset dihitung dengan:

address = (<segment> \* 16) + <offset>

Program Memori- program dapat ditempatkan di manapun di dalam memori. perintah jump and call dapat digunakan untuk menyingkat lompatan di dalam segmen kode 64 KB, seperti halnya untuk lompatan jauh di manapun di dalam memori 1 MB. Seluruh perintah lompatan yang bersyarat dapat digunakan untuk melompat sekitar + 127 - - 127 bytes dari instruksi yang ada.

Memori data - prosesor dapat mengakses data di tiap orang lebih dari 4 segmen yang tersedia, yang membatasi ukuran dari memori yang dapat diakses sampai 256 KB ( jika seluruh empat segmen menunjuk pada 64 KB blok berbeda). Mengakses data dari Data, Code, segmen Extra atau Stack biasanya dapat dilaksanakan dengan awalan perintah DS:, CS:, SS: atau ES: ( beberapa register dan instruksi dengan tak hadir boleh gunakan segmen SS atau ES sebagai ganti segmen DS).

Set Instruksi

Set instruksi 8088  terdiri dari perintah-perintah berikut:

 \* Instruksi perpindahan data.

 \* Aritmatika - penjumlahan, pengurangan, penaikan, penurunan, mengkonversi byte/word dan pembandingan.

 \* Logika - DAN, OR, eksklusif OR, shift/rotate dan test.

 \* Manipulasi string - load, store, move, compare dan scan untuk byte/word.

 \* Kontrol transfer - conditional, unconditional, panggilan subroutine dan kembali dari subroutine.

 \* Perintah Input/Output.

 \* Lain-lain - setting/clearing flag bits, stack operations, software interrupts, dan lain-lain.

KESIMPULAN

8088 adalah desain hybrid 8/16-bit: 16-bit internal, dengan 8-bit I/O. ini berarti bahwa Sistem designer dapapt menggunakan chip pendukung 8-bit yang murah dan tersedia. Sehingga menurunkan harga sebuah sistem komputer. Kemampuannya tidak begitu hebat, mesin-mesin Z-80 dan 8085 kadang-kadang lebih unggul, tetapi 8088 terjual cukup banyak. Apalagi setelah divisi IBM memilihnya sebagai prosesor utama IBM PC. 8088 menjadi sejarah dan terjual jutaan unit, rekor kedua setelah Z-80.

Sayangnya Intel membuat beberapa keputusan yang buruk dalam desain. Pertama, 8088 menggunakan arsitektur bersegmen. Kedua untuk alasan yang tidak efisien, Intel memutuskan untuk membatasi akses base-memory dalam suatu cara menjadi hanya 640K. Meskipun pada masa itu keputusan ini adalah untuk jangka pendek semua user prosesor x86 hingga saat ini merasa kesulitan dengan pembatasan 640K tersebut. Ini dapat ditemukan jika user menerima pesan kesalahan 'out of memory'. DOS, Windows, dan bahkan Windows 95 memiliki permasalahan base-memory yang disebabkan oleh batas 640K. Hanya sistem operasi murni 32-bit seperti OS/2 dan Windows NT yang tidak memiliki masalah ini. Saat itu user harus menunggu 7 tahun sebelum kemunculan 386. <b>[muchad 2008]</b>

**(Set Instruksi ARM)**

ARM

- Advanced RISC Machines (ARM) limited telah mendesain suatu famili mikroprosesor dan melisensikan desain tersebut ke perusahaan lain untuk fabrikasi chip yang penggunaannya dalam produk komputer dan sistem uang embedded.

- Perusahan ARM yang relatif baru, merupakan perkembangan dari perusahan Acorn Computer yang mengembangkan desain prosesor pada awal tahun 1980-an.

- Penggunaan utama mikroprosesor ARM adalah pada aplikasi embedded yang berdaya rendah dan berbiaya rendah, seperti misalnya mobile telephone, modem komunikasi, sistem manejemen mesin mobil, dan hand-held

**3.1 Register, Akses Memori dan Transfer Data**

Dalam arsitektur ARM, memori adalah byte-addressble, menggunakan alamat 32-bit, dan register prosesor memiliki panjang 32-bit. Dua panjang operand digunakan dalam memindahkan data antara memori dan register prosessor : byte (8 bit) dan word (32 bit).

**3.1.1 Struktur Register**

**- Register prosesor yang digunakan oleh program aplikasi** ditampilkan terdapat enam belas register 32 bit berlabel R0 hingga R15 yang terdiri dari lima belas general-purpose register (R0 hingga R14) dan register program counter (PC), R15, yang terdiri dari 15.

- General purpose register dapat menyimpan alamat memori atau operand data. Current progarm status register (CPRS), atau cukup disebut register status, menyimpan condition code flag (N, Z, C, V), interupt disable flag, dan bit mode prosesor.

- Disini kita akan mengasumsikan bahw prosesor dalam mode user dan dieksekusi sebagai program aplikasi.terdapat 15 general purpose register tambahn yang yang disebut register banked.

**3.1.2 Instruksi Akses Memory Dan Mode Pengalamatan**

**- Eksekusi Conditional Instruksi**

Fitur yang membedakan dan agak tidak biasa dari prosesor ARM adalah semua instruksinya dieksekusi secara conditional, tergantung pada kondisi yang ditetapkan pada instruksi tersebut. Instruksi tersebut dieksekusi hanya jika keadaan saat ini dari conditional code flag prosesor memenuhi kondisi yang ditetapkan dalam bit b31-28 dari instruksi tersebut. Jika tidak prosesor melanjutkan ke instruksi berikutnya salah satu kondisi tersebut digunakan untuk mengindikasikan bahwa instruksi tersebut selalu dieksekusi

**- Mode Pengalamatan Memori**

Metode dasar untuk mengalamati operand memori adalah membangkitkan effective address, EA, dari operan tersebut dengan menambahkan offset bertanda keisi base register Rn, yang ditentukan dalam instruksi. Besarnya offset tersebut dapat berupa nilai immediate yang terdapat dalam 12 bit low order instruksi atau isi dari register ketiga, Rm, yang dinamai dengan 4 bit low order tanda arah offset terdapat dalam field OP-code.

- OPERAND LOAD/STORE MULTIPLE

Selain instruksi load dan store untuk operand tunggal, terdapat 2 instruksi untuk me-load dan menyimpan banyak operand. Instruksi itu disebut instruksi transfer block. Sub set apapun dari general purpose register load atau disimpan. Hanya operand word yang diperbolehkan, dan OP code yang digunakan dalam load multiple dan store multiple. Operand memori harus berada dalam lokasi word yang berurutan.

**3.1.3 Instruksi Move Register**

Acap kali kita perlu meng-copy isi satu register ke register lain atau untuk me-load nilai immediate ke suatu register. Instruksi move

MOV Rd , Rm

**3.2 Instruksi Aritmatika Dan Logika**

Set instruksi ARM memiliki sejumlah instruksi untuk operand aritmatika dan logika pada operand yang berada dalam general-purpose register atau dinyatakan sebagai operand immediate dalam instruksi itu sendiri.

Terdapat instruksi untuk operand logika AND,OR,NOT,XOR, dan bit-clear. Instruksi seperti compare disediakan untuk men-set condition code flag berdasarkan hasil dari operasi aritmatika dan logika pada dua operand

**3.2.1 Instruksi Aritmatika**

Ekspresi bahasa assembly dasar untuk instruksi aritmatika adalah

Opcode Rd, Rn, Rm

Dimana operasi yang ditetapkan oleh OP code dilakukan menggunakan operand dalam general-purpose register Rn dan Rm. Hasilnya diletakkan dalam register Rd. Misalnya, instruksi

ADD R0, R2, R4

Menjalankan operasi

R0  [R2] + [R4]

Dan instruksi

SUB R0, R6, R5

Menjalankan operasi

R0  [R6] – [R5]

**3.2.2 Instruksi Logika**

Operasi logika AND, OR, XOR, dan Bit-clear diimplementasikan oleh instruksi OP code AND, ORR, EOR, dan BIC. Kode tersebut memiliki format yang sama dengan instruksi aritmatika. Instruksi

AND Rd, Rn, Rm

Menjalankan operasi

Rd  [Rn] ^ [Rm]

Yang merupakan bitwise logical AND antara operand dalam register Rn dan Rm. Misalnya, jika register R0 berisi pola hexadesimal 02FA62CA dan R1 berisi pola 0000FFFF, maka instruksi

AND R0, RO, R1

Akan menyebabkan pola 000062CA diletakkan dalam register R0.

**3.3 Instruksi Branch**

Instruksi branch CONDITIONAL berisi offset 24-bit, 2’-complement, bertanda yang ditambahkan ke isi ter-update Program Counter untuk menghasilkan alamat target branch.

Instruksi Branch dieksekusi dengan cara yang sama seperti instruksi ARM yang lain, yaitu dieksekusi hanya jika keadaan terbaru condition code flag berhubungan dengan kondisi ditetapkan dalam field codition instruksi tersebut.

**3.3.1 Setting Condition Code**

beberapa instruksi, seperti compare, dinyatakan sebagai berikut

CMP Rn, Rm

Yang menjalankan operasi

[Rn] – [Rm]

Memiliki tujuan utama untuk men-set condition code flag berdasar pada hasil operasi pengurangan.

**3.3.2 Program Loop untuk penambahan bilangan**

Operasi load dan store dilakukan oleh instruksi pertama, kedua, dan terakhir yang digunakan oleh mode pengalamatan relative. Ini mengasumsikan bahwa lokasi memori N, pointer, dan SUM terdapat dalam rentang yang terjangkau oleh offset relatif terhadap PC. Lokasi memori pointer berisi alamat NUM 1 dari bilangan pertama yang akan ditambahkan, N berisi jumlah entri didalam list dan SUM digunakan untuk menyimpan jumlah tersebut

**3.4 Bahasa Assembly**

Bahasa assembly ARM memiliki assembler directive untuk menyiapkan ruang penyimpanan, menetapkan nilai numerik ke label alamat dari simbol konstanta, menentukan dimana program dan blok data akan ditempatkan dalam memori, menetapkan akhir teks source program fasilitas tersebut didekskripsikan secara umum.

**3.5 Kontrol Aliran Program**

- Condition Code Flag

68000 memiliki lima condition code flag, disimpan dalam register status. Selain flag M,Z,V, dan C yang dideskripsikan pada bagian 2.4.6, 6800 memiliki lima flag, X (extend). Di-set dengan cara yang sama dengan flag C, tetapi tidak dipengaruhi oleh banyak instruksi.

- Instruksi Branch

Instruksi conditional branch menyebabkan eksekusi program berlanjut dengan instruksi pada alamat target branch jika kondisi branch dipenuhi.

**3.6 Operasi I/O**

Prosessor 68000 mensyaratkan semua status dan buffer data dalam antar muka perangkat I/O menjadi addressable seperti lokasi memori. Ini berarti program-program I/O dalam komputer 68000 dapat dicapai.

**3.7 Stack dan Subroutine**

Suatu stack dapat diimplementasikan, menggunakan register alamat apapun sebagai pointer. Mode pengalamatan Autoincrement dan Autodecrement memfasilitasi proses ini. Satu register spesifik, register A7, ditunjuk sebagai pointer stack prosesor, dan stack yang ditunjuk register ini disebut stack prosesor. Ini adalah stack yang digunakan dalam semua operasi stack yang dilakukan prosesor secara otomatis, sebagaimana dalam kasus linkage subroutine.

**3.8 Instruksi Logika**

Operand yang terlibat dalam instruksi ini memiliki panjang tetap 32, 16, atau 8 bit. Pada beberapa aplikasi perlu untuk memanipulasi: ukuran data yang lain, mungkin hanya bit individu, dan melakukan operasi logika pada data ini.

**3.9 Register dan Pengalamatan**

Dalam arsitektur 1A-32, memori adalah byte addressable menggunakan alamat 32-bit, dan instruksi beroperasi pada operand data 8 dan 32 bit. Ukuran operand ini disebut byte dan doubleword dalam istilah intel.

**3.9.1 Struktur Register 1A-32**

Terdapat delapan floating-point register untuk menyimpan operand data floating point doubleword atau quadword (64 bit). Floating-point register yang memiliki field ekstensi untuk menyediakan panjang total 80 bit.

Arsitektur 1A-32 berbasis pada model memori yang menghubungkan area yang berbeda di dalam memori, yang disebut segmen dengan kegunaan yang berbeda.

**3.9.2 Mode Pengalamatan 1A-32**

Arsitektur 1A-32 memiliki set mode pengalamatan yang besar dan fleksibel. Mode tersebut didesain untuk mengakses item data individu atau item data yang merupakan aggota dari list yang berurutan yang mulai pada alamat memori tertentu.

Mode dasar, yang tersedia pada kebanyakan prosesor telah dideskripsikan. Mode tersebut adalah : Immediate, Absolute, Register, dan Register indirect.