Set Instruksi:

Set instruksi ?

- Operasi dari CPU ditentukan oleh instruksi-instruksi yang dilaksanakan atau dijalankannya. Instruksi ini sering disebut sebagai instruksi mesin (*machine instructions*) atau instruksi komputer (*computer instructions*).
- Kumpulan dari instruksi-instruksi yang berbeda yang dapat dijalankan oleh CPU disebut set Instruksi (*Instruction Set*).

Elemen Instruksi

- Operation code (Op code)
  - Kerjakan, **menentukan operasi yang akan dilaksanakan**
- Source Operand reference
  - Dengan data ini, **merupakan input bagi operasi yang akan dilaksanakan**
- Result Operand reference
  - Simpan hasilnya kesini, **merupakan hasil dari operasi yang dilaksanakan**
- Next Instruction Reference
  - Setelah selesai, kerjakan ini ... , **memberitahu CPU untuk mengambil (*fetch*) instruksi berikutnya setelah instruksi yang dijalankan selesai.**

Operands ?

- Ingat: Semua instruksi dijalankan dalam CPU
- Rata-rata operasi hanya membutuhkan register sebagai tempat membaca /menyimpan operand
- Adakalanya juga register tidak berisi operand tapi menunjuk ke tempat penyimpanan lainnya (memory, cache, modul I/O)
- Source dan results operands dapat berupa salah satu di antara tiga jenis berikut :
  - Main memory (or virtual memory or cache)
  - CPU register
  - I/O device

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

Penyajian Instruksi

- Dlm kode mesin setiap instruksi memiliki pola-bit tertentu yang unik
- Agar dapat dimengerti manusia, dibuatlah representasi simbolik instruksi, biasanya berupa singkatan (disebut *mnemonic*)
  - Contoh: ADD, SUB, LOAD
- Sedangkan Operand juga disajikan secara simbolik
  - Contoh: ADD **A, B**

Jenis Instruksi

- Data processing, ***Arithmetic dan Logic Instructions***
- Data storage (main memory), ***Memory instructions***
- Data movement (I/O), ***I/O instructions***
- Program flow control, ***Test and branch instructions***

Jumlah addres (a)

- 3 addres
  - Bentuk umum: [OPCODE] [AH], [AO1], [AO2]
  - Satu alamat hasil, dua alamat operand
  - Misal: **SUB Y, A, B**
    - Bentuk algoritmik: Y A – B
    - Arti: Kurangkan isi Reg A dengan isi Reg B, kemudian simpan hasilnya di Reg Y.
  - Bentuk ini tidak umum digunakan di komputer
  - Mengoperasikan banyak register sekaligus
  - Program lebih pendek

Jumlah addres (b)

- 2 addres
  - Bentuk umum: [OPCODE] [AH], [AO]
  - Satu alamat hasil merangkap operand, satu alamat operand
  - Misal: **SUB Y, B**
    - Bentuk algoritmik: Y Y – B
    - Arti: Kurangkan isi Reg Y dengan isi Reg B, kemudian simpan hasilnya di Reg Y.
  - Bentuk ini masih digunakan di komputer sekarang
  - Mengoperasikan lebih sedikit register, tapi panjang program tidak bertambah terlalu banyak

Jumlah addres (c)

- 1 addres
  - Bentuk umum: [OPCODE] [AO]
  - Satu alamat operand, hasil disimpan di accumulator
  - Misal: **SUB B**
    - Bentuk algoritmik: AC AC – B
    - Arti: Kurangkan isi Acc. dengan isi Reg B, kemudian simpan hasilnya di Acc.
  - Bentuk ini digunakan di komputer jaman dahulu
  - Hanya mengoperasikan satu register, tapi program menjadi bertambah panjang

Jumlah addres (d)

- 0 (zero) addres
  - Bentuk umum: [OPCODE] [O]
  - Semua alamat operand implisit, **disimpan dalam bentuk stack**. Operasi yang biasanya membutuhkan 2 operand, akan mengambil isi stack paling atas dan di bawahnya
  - Misal: **SUB**
    - Bentuk algoritmik: S[top] □ S[top-1] – S[top]
    - Arti: Kurangkan isi Stack no.2 dari atas dengan isi Stack paling atas,kemudian simpan hasilnya di Stack paling atas
  - Ada instruksi khusus Stack: **PUSH** dan **POP**
  - Contoh lain:

    push a
    push b
    add
    pop c
  - Berarti:

    c = a + b

Contoh Format Instr 3 Alamat

- 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

Contoh Format Instr 2 Alamat

- 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

Contoh Format Instr 1 Alamat

- 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

Contoh Format Instr 0 Alamat

- 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

Addres banyak ? Sedikit ?

- Addres banyak
  - Instruksi semakin kompleks
  - perlu register banyak
  - Program lebih pendek
  - Lebih cepat ?
- Addres sedikit
  - Instruksi lebih sederhana
  - Eksekusi lebih cepat

Pertimbangan Perancangan (1)

- Semakin banyak register yang diolah dalam satu instruksi → semakin lambat
- Semakin banyak baris operasi untuk mengeksekusi sebuah program → juga semakin lambat
- Oleh karena itu, CPU masakini bisa dikatakan menggunakan perpaduan dari beberapa format instruksi, tergantung konteksnya

Pertimbangan Perancangan (2)

- Register
  - Jumlah register yang tersedia dlm CPU
  - Operasi apa yg dpt dikerjakan oleh masing-masing registers?
- Addressing modes (…)

Jenis Operand

- Addres
- Number
  - Integer/floating point
- Character
  - ASCII etc.
- Logical Data
  - Bits or flags

Jenis Operasi

- Data Transfer
- Arithmetic
- Logical
- Conversion
- I/O
- System Control
- Transfer of Control

Data Transfer

- menentukan
  - Source, Destination → lokasi operand sumber dan operand tujuan.
    Menetapkan Lokasi-lokasi tersebut dapat berupa memori, register atau bagian paling atas daripada stack.
  - Jumlah data → 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 :

  Menetapkan alamat memori.

  Menjalankan transformasi alamat memori virtual ke alamat memori aktual.

  Mengawali pembacaan / penulisan memori

Data Transfer

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

- Add, Subtract, Multiply, Divide
- Signed Integer
- **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

- Bitwise operations
- AND, OR, NOT
- **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.

Conversion

- Contoh: Biner ke Decimal
- **Operasi set instruksi untuk conversi :**

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

Input/Output

- Tersedia instruksi khusus
- Atau digunakan instruksi data movement (memory mapped)
- Atau dikerjakan oleh controller (DMA)
- **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

Systems Control

- Privileged instructions
- CPU harus berada pada state tertentu
  - Ring 0 pada 80386+
  - Kernel mode
- Digunakan oleh operating systems
- 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.

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

Transfer Control

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.

Transfer Control

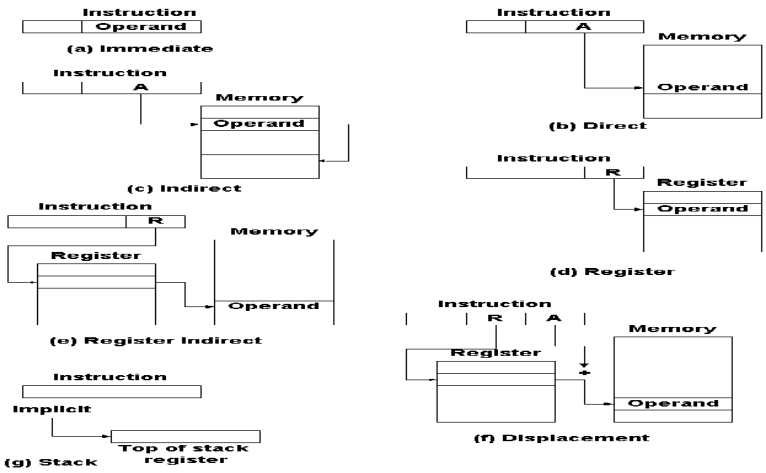
- Branch
  - Contoh: branch to x if result is zero
- Skip
  - Contoh: increment and skip if zero
  - ISZ Register1
  - Branch xxxx
  - ADD A
- Subroutine call
  - interrupt call

ADDRESSING MODES

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

- ⌘ *Immediate*
- ⌘ *Direct*
- ⌘ *Indirect*
- ⌘ *Register*
- ⌘ *Register Indirect*
- ⌘ *Displacement*
- ⌘ *Stack*

Tabel Basic Addressing Modes

Mode	**Algorithm**	**Principal Advantage**	**Principal Disadvantage**
Immediate	Operand = A	No memory reference	Limited operand magnitude
Direct	EA = A	Simple	Limited address space
Indirect	EA = (A)	Large address space	Multiple memory references
Register	EA = R	No memory Reference	Limited address space
Register Indirect	EA = (R)	Large address space	Extra memory reference
Displace-ment	EA=A+(R)	flexibility	Complexity
Stack	EA=top of Stack	No memory Reference	Limited applicability

Gambar Addressing Mode