MODUL PRAKTIKUM

SISTEM MIKROPROSESOR

LABORATORIUM SISTEM PENGATURAN DAN KOMPUTER

DEPARTEMEN TEKNIK ELEKTRO

FAKULTAS TEKNIK

UNIVERSITAS SUMATERA UTARA

2013

DAFTAR ISI

BAB I

PENGENALAN MIKROPROSESOR 8086

I.1 PENGENALAN 8086

Intel 8086 merupakan mikroprosesor 16-bit dengan arsitektur x86 yang di desain oleh Intel antara tahun 1976 dan pertengahan 1978, pada saat dikeluarkan. Dari segi arsitektur secara umum setiap mikroprosesor dapat dipandang oleh beberapa komponen, yaitu:

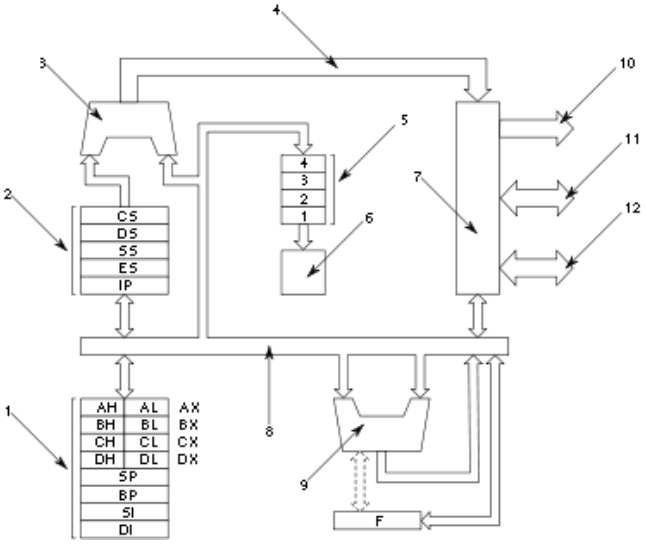
- ALU (Arithmetic Logic Unit) merupakan unit yang melakukan semua operasi aritmatika dan logika.
- Register kerja yang dapat dijangkau pemakai/pemrogram.
- Register prosesor yang dapat digunakan hanya oleh mikroprosesor secara implisit dan tak dapat dijangkau pemakai.
- Kendali dan pewaktu.

I.1.1 ARSITEKTUR 8086

Secara umum arsitektur pada mikroprosesor 8086/8088 dapat dilihat pada gambar 1.

Beberapa bagian pada gambar dijelaskan sebagai berikut:

1=register utama; 2=register

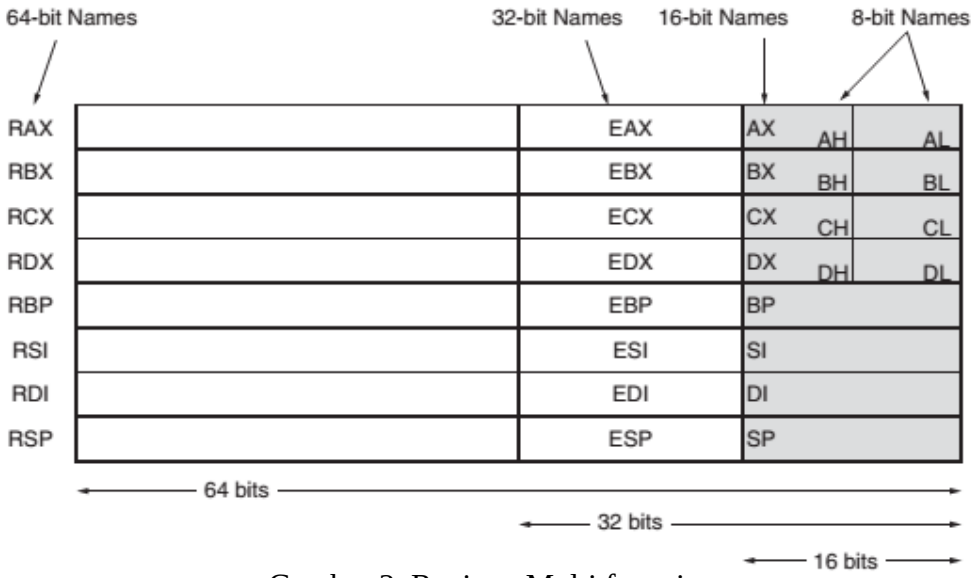


Gambar 1. Arsitektur 8086/8088

segmen dan IP; 3=penambah alamat; 4=bus alamat internal; 5=antrian instruksi; 6=unit kontrol; 7=bus antarmuka; 8=databus internal; 9=ALU (Arithmatic Logika Unit); 10/11/12=bus alamat/data/kontrol eksternal.

I.1.2 REGISTER

I.1.2.1 REGISTER MULTI FUNGSI



Gambar 2. Register Multi fungsi

Pada gambar 2 dapat dilihat beberapa register yang ada pada mikroprosesor 8086.Dan berikut adalah penjelasan singkat masing – masing register.

- Register A (AL,AH,AX,EAX,RAX) Akumulator

  Register A adalah akumulator.dimana akumulator biasanaya digunakan oleh instruksi yang melakukan perkalian, pembagian, penjumlahan dan pengurangan. AX adalah register berukuran 16 bit dimana dibagi menjadi 2 register AH (8 bit high) dan AL (8 bit low). RAX dan EAX memiliki ukuran 64 bit dan 32 bit.

- Register B (BL,BH,BX,EBX,RBX) Base

  Register B (Base) terkadang menyimpan alamat offset dari sebuah lokasi memori didalam sistem. Register Base juga dapat digunakan untuk mengalamatkan data memori.

- Register C (CL,CH,CX,ECX,RCX) Count

  Register C juga merupakan register multi-fungsi yang banyak digunakan oleh banyak instruksi.Biasanya register Count digunakan untuk pencacahan perluangan suatu subrutin ataupun label.

- Register D (DL,DH,DX,EDX,RDX) Data

  Register Data digunakann untuk penyimpanan sebagian dari hasil operasi perkalian ataupun pembagian

- RBP (Base Pointer)

  Base Pointer digunakan untuk menunjukan lokasi suatu memori untuk transfer data memori.

- RSI (Source Index)

  Source Index digunakan untuk mengalamatkan sumber data string untuk operasi string.

- RDI (Destination Index)

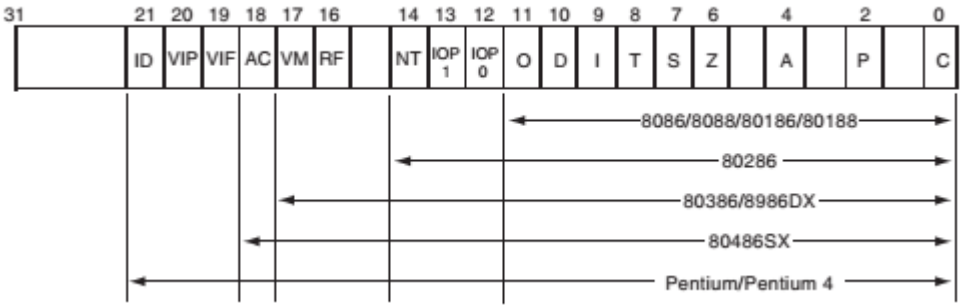
  Destination Index digunakan untuk mengalamatkan tujuan data string untuk operasi string.

- RSP (Stack Pointer)

  Stack Pointer mengalamatkan area memori yang disebut stack. Memori stack menyimpan data melalui pointernya

I.1.2.2 Register Flags

Register Flags adalah register yang mengindikasikan kondisi mikroprosesor dan kontrol operasinya. Register Flag hanya bersifat penanda (bendera) dan tidak merubah nilai yang ada pada register lain dan memori. Pada gambar 3 adalah register flag yang digunakan oleh mikroprosesor 8086.



Gambar 3. Register Flag

- Carry

  Flag Carry menampung carry dari penjumlahan atau borrow dari pengurangan. Flag carry juga dapat mengindikasikan error.

- Parity

  Flag Parity digunakan unutk mengecek kesalahan bilangan. Logika memiliki logika 0 untuk parity ganjil dan logika 1 untuk parity genap.

- Auxiliary Carry

  Auxiliary Carry juaga digunakan untuk mengecek carry (half carry)

- Zero

  Flag Zero digunakan untuk menandakan bahwa hasil dari operasi aritmatika bernilai nol.

- Sign

  Flag sign digunakan untuk menandai tanda aritmatika dari hasil operasi penjumlahan ataupun hasil logika.

- Interupt

  Interupt digunakan untik mengontrol operasi INTR(permintaan interupt).

- Trap

  Trap digunakan untuk mengenablekan fitur debugging on-chip.

- Overflow

  Overflow terjadi ketika bilangan bertanda dijumlahkan ataupun dikurangkan.

- Direction

  Direction digunakan untuk penanda pada register DI atau SI arah penambahannya bersifat maju atau mundur.

I.1.3 Instruksi

Instruksi yang akan dibahas pada modul ini adalah instruksi perpindahan data.

I.1.3.1 Instruksi Umum (Perpindahan Data)

Instruksi perpindaahan data merupakan instruksi yang paling sering dalam permrograman mikroprosesor dan juaga relatif lebih mudah untuk dipahami.

- MOV

  MOV merupakan instruksi paling umum dalam pemrograman mikroprosesor. Instruksi MOV melaksanakan operasi perpindahan data . Contoh : MOV AX,BX. MOV BX,678h

- PUSH / POP

  PUSH dan POP adalah instruksi perpindahan data yang spesifik pada segmen stack. Instruksi PUSH berfungsi untuk mengamankan nilai suatu register kedalam memori stack sebelum mikroprosesor memasuki suatu rutin ataupun prosedur, sedangkan Instruksi POP digunakan untuk mengembalikan nilai register dari memori stack setelah rutin selesai dijalankan.

- LEA (Load Effective Addres)

  Instruksi LEA digunakan untuk mengisi register 16 bit manapun dengan alamat offset. Contoh :

  DATA1 DW 2000H ;define DATA1

  .CODE ;start code segment

  .STARTUP ;start program

  LEA SI,DATA1

I.1.3.2 Instruksi Aritmatika dan Logika

- ADDITION,SUBTACTION, COMPARISON

  Beberapa instruksi yang digunakan adalah ADD, SUB dan CMP.

- MULTIPLICATION, DIVISION

  Instruksi yang digunakan untuk operasi perkailain dan pembagian adalah MUL dan DIV.

- BASIC LOGIC

  Beberapa instruksi logika pada mikroprosesor adalah AND, OR, XOR, NOT dan TEST.

- SHIFT , ROTATE

  Instruksi SHIFT dan ROTATE digunakan untuk memanipulasi bilangan pada level bit binernya , sama dengan instruksi logika dasar.

I.1.3.3 Instruksi Kontrol Program

- JUMP

  Instruksi kontrol program utama adalah JUMP , instruksi inin memungkinkan mikroprosesor "melompati" bagian program menuju bagian lain dari memori untuk instruksi berikutnya.

  Instruksi JUMP sendiri terbagi 2 yaitu CONDITIONAL JUMP dan UNCONDITIONAL JUMP.

I.1.3.4 Interupsi

Interupsi jika diartikan adaalh "panggilan" mendesak dari hardware(biasanys berasal dari sinyal hardware) ataupun "panggilan" yang dihasilkan dari exekusi sofware.(biasanya akibat dari beberapa event internal).

I.1.4 PENGALAMATAN/ *ADDRESSING*

Karena Instruksi MOV sangat umum dan flexibel . Instruksi MOV menyediakan penjelasan dasar dari mode pengalamatan data,

I.1.4.1 PENGALAMATAN LANGSUNG/ *DIRECT ADDRESSING*

Pengalamatan langsung memindahkan sebuah byte atau word antara lokasi memori dan register . Pengalamatan ini tidak mendukung transfer memori ke memori. Contoh : MOV AX,[1204h]

I.1.4.2 PENGALAMATAN REGISTER/ *REGISTER ADDRESSING*

Pengalamtan register mentransfer salinan byte atau word data dari register sumber atau isi dari lokasi suatu memori ke register tujuan ataupun lokasi memori tujuan . Contoh : MOV AX,BX

I.1.4.2 PENGALAMATAN SEGERA/ *IMMEDIATELY ADDRESSING*

Pengalamatan segera mentransfer sebuah data byte atau word ke tujuan register atau lokasi memori. Contoh : MOV AL,22h

BAB II

II.1 PERCOBAAN 1: PENJUMLAHAN 10 BUAH DATA

```
name "jumlah sepuluh data"
org 100h          ; set alamat mulai 100h
mov cx, 10        ; set
mov al, 0
mov bx, 0

; jumlahkan semua variabel
next:
add al, vector[bx] ;jumlahkan semua bilangan

; Byte berikutnya
inc bx

; ulangi sampai cx = 0:
loop next

; simpan hasil di m
mov m, al

print:
mov ah, 2   ; Fungsi print.

mov dl, 0ah ;.
int 21h
mov dl, 0dh ;.
int 21h


; print hasil dalam desimal
mov al, m
call print_al


; wait for any key press:
mov ah, 0
int 16h
```

```
ret

; variabel
vector db 5, 4, 5, 2, 1, 6, 5, 8, 9, 4
m db 0

print_al proc
cmp al, 0
jne print_al_r
    push ax
    mov al, '0'
    mov ah, 0eh
    int 10h
    pop ax
    ret
print_al_r:
    pusha
    mov ah, 0
    cmp ax, 0
    je pn_done
    mov dl, 10
    div dl
    call print_al_r
    mov al, ah
    add al, 30h
    mov ah, 0eh
    int 10h
    jmp pn_done
pn_done:
    popa
    ret
endp
```

II.2 PERCOBAAN 4: HELLO WORLD

```
    name "hi-world"
    org 100h       ; set
    mov     ax, 0b800h
    mov     ds, ax

    mov [02h], 'H'

    mov [04h], 'e'

    mov [06h], 'l'

    mov [08h], 'l'

    mov [0ah], 'o'

    mov [0ch], ','

    mov [0eh], 'W'

    mov [10h], 'o'

    mov [12h], 'r'

    mov [14h], 'l'
mov [16h], 'd'

mov [18h], '!'

; Warnai semua karakter
mov cx, 12  ; jumlah karakter
```

```
mov di, 03h ; Mulai dari byte setelah ‘H’

c:  mov [di], 11101100b   ; Merah (1100) diatas kuning(1110)
    add di, 2 ;
    loop c                    ;Ulangi sebanyak 12 kali

ret
```

BAB II.3 PROSEDUR CALL

```
    call a1
    call a2
    ret

    a1:
    mov ax, 1
    a2:
    add ax, 3
    ret
```

BAB II.4 Menampilkan bilangan dalam biner

```
        name "add-sub"

        org 100h

        mov al, 5       ; bin=00000101b
        mov bl, 10      ; hex=0ah or bin=00001010b

        add bl, al          ;jumlahkan bl dan al , lalu simpan di bl

        ; 15 - 1 = 14 (decimal) or hex=0eh or bin=00001110b
```

```
sub bl, 1             :kurangkan bl dengan 1 lalu simpan di bl

; print hasil dalam biber
mov cx, 8
print: mov ah, 2   ; print function.
      mov dl, '0'
      test bl, 10000000b  ; tesbit pertama
      jz zero
      mov dl, '1'
zero:  int 21h
      shl bl, 1
loop print

; print binary suffix:
mov dl, 'b'
int 21h


ret
```