MODUL III

FLOW CONTROL INSTRUTIONS

Intruksi lompatan (jump) dan perulangan (loop) digunakan untuk pengambilan keputusan dan mengulang bagian-bagian kode. Untuk mengontrol instruksi lompatan (jump) dan perulangan (loop) dapat dilakukan dengna terlebih dahulu menyeleksi kondisi logika (conditional jump) dan tanpa menyeleksi kondisi (unconditional jump).

Unconditional Jump

Instruksi JMP (jump) menyebabkan transfer kontrol tak bersyarat (*unconditional jump*) dan akan melompati daerah data program tertentu. Sintaksnya adalah:

JMP tujuan

Di mana label tujuan umunya adalah label dalam suatu segmen yang sama seperti JMP itu sendiri.

```
org     100h
mov     ax, 5           ; ax = 5.
mov     bx, 2           ; bx = 2.
jmp     hitung            ; lompat ke 'hitung'.
lagi:   jmp stop        ; go to 'stop'.
hitung:
add     ax, bx          ; ax = ax+bx.
jmp     lagi            ; lompat ke 'lagi'.
stop:
ret                     ; selesai
```

Compare

Instruksi CMP (compare) digunakan untuk membandingkan 2 buah operand, dengan sintaks :

CMP tujuan,sumber

Instruksi ini membandingkan sumber dan tujuan dengan cara mengurangkan tujuan dengan sumber. Perintah CMP tidak mempengaruhi nilai tujuan dan sumber, perintah ini akan mempengaruhi flag register (OF, SF, ZF, CF) sebagai hasil perbandingan. Operand CMP tidak dapat membandingkan antar 2 lokasi memori. Perintah CMP biasanya diikuti dengan perintah lompat yang melihat keadaan flag register.

Adapun flag-flag yang terpengaruh oleh perintah CMP ini adalah:

- OF akan 1, jika operand1 lebih kecil dari operand2 pada operasi bilangan bertanda.
- SF akan 1, bila operand1 lebih kecil dari operand2, pada operasi bilangan bertanda.
- ZF akan 1, jika operand1 nilainya sama dengan operand2.
- CF akan 1, jika operand1 lebih kecil dari operand2 pada operasi bilangan tidak bertanda.

Perlu anda ingat bahwa CMP tidak dapat membandingkan antar 2 lokasi memory.

```
org 100h
mov ah,9
lea dx,input
int 21H
mov ah,1
int 21h
mov a,al
mov ah,9
lea dx,enter
int 21h
lea dx,input1
int 21h
mov ah,1
int 21h
mov bl,al
mov ah,9
lea dx,enter
int 21h
cmp a,bl
je equal
jl less
jg greater
jmp  stop
equal:
    mov ah,9
    lea dx,sama
    int 21h
    jmp stop
less:
    mov ah,9
    lea dx,kurang
    int 21h
    jmp stop
greater:
    mov ah,9
    lea dx,lebih
    int 21h
stop:
    ret
a db ?
b db ?
input db "masukkan nilai A=
$"
input1 db "masukkan nilai
B= $"
kurang db "A < B $"
sama db "A = B $"
lebih db "A > B $"
enter db ,0dh,0ah,'$'
```

Conditional Jump

Instruksi conditional jump memindahkan kendali ke alamat tujuan ketika kondisi flag bernilai benar. Sintaknya sebagai berikut :

JCond tujuan

Alamat tujuan harus antara –128 - +127 byte dari lokasi sekarang. Cond mengacu pada kondisi flag, mengidentigikasikan satu atau lebih keadaan.

Flag diset oleh instruksi aritmetik, perbandingan dan boolean. Setiap instruksi jump kondisional memeriksa satu atau beberapa flag, mengembalikan hasil benar atau salah. Jika hasilnya benar, maka jump dilakukan, sebaliknya jika salah maka tidak ada yang dilakukan dan instruksi selanjutnya akan dilakukan.

Contoh :

```
ORG 100h
Proses:
MOV AH,1 ; Servis untuk mengecek buffer keyboard
INT 16h ; Laksanakan !
JNZ EXIT ; Jika ada tombol yang ditekan, lompat Ke EXIT
MOV AH,09 ; Servis untuk cetak kalimat
LEA DX,Kal0 ; Ambil alamat efektif Kal0
INT 21h ; Cetak kalimat !
JMP Proses ; Lompat ke Proses
exit:
INT 20h ; Selesai
RET
Kal0 DB 'Tekan sembarang tombol untuk berhenti ! '
DB 13,10,'$'
END
```

Looping

Loop adalah suatu rangkaian instruksi yang diulang. Banyaknya pengulangan mungkin dapat diketahui akhirnya, atau tergantung pada kondisi tertentu.

Bentuknya sbb:

LOOP label_tujuan

Pencacah untuk loop adalah register CX yang diinisialisasi ke loop_count. Ekesekusi instruksi LOOP menyebabkan CX turun (decremented) secara otomatis, dan jika CX tidak nol, maka control menstransfernya ke label_tujuan. Jika CX = 0, instruksi berikutnya setelah LOOP yang dikerjakan. Label_tujuan harus berada sebelum instuksi LOOP dengan tidak lebih dari 126 byte. Berikut ialah salah satu contoh pengaplikasian looping dalam mencetak beberapa karakter berbeda dalam hal ini huruf .

```
ORG 100h
Proses :
MOV AH,02h ; Nilai servis
MOV DL,'A' ; DL=karakter 'A' atau DL=41h
MOV CX,10h ; Banyaknya pengulangan yang akan
Ulang:
INT 21h ; Cetak karakter !!
INC DL ; Tambah DL dengan 1
LOOP Ulang ; Lompat ke Ulang
INT 20h ; selesai
RET
END
```

Perintah Lompat		Kondisi
JA	<Jump If Above>	Lompat, jika Operand1 > Operand2 untuk bilangan tidak bertanda
JG	<Jump If Greater>	Lompat, jika Operand1 > Operand2 untuk bilangan bertanda
JE	<Jump If Equal>	Lompat, jika Operand1 = Operand2
JNE	<Jump If Not Equal>	Lompat, jika Operand1 tidak sama dengan Operand2
JB	<Jump If Below>	Lompat, jika Operand1 < Operand2 untuk bilangan tidak bertanda
JL	<Jump If Less>	Lompat, jika Operand1 < Operand2 untuk bilangan bertanda
JBE	<Jump If Below or Equal>	Lompat, jika operand1 <= Operand2 untuk bilangan tidak bertanda
JLE	<Jump If Less or Equal>	Lompat, jika Operand1 <= Operand2 untuk bilangan bertanda
JAE	<Jump If Above or Equal>	Lompat, jika Operand1 >= Operand2 untuk bilangan tidak bertanda
JGE	<Jump If Greater or Equal>	Lompat, jika Operand1 >= Operand2 untuk bilangan bertanda
JC	<Jump Carry>	Lompat, jika Carry flag=1

JCXZ <Jump If CX is Zero>	Lompat, jika CX=0
JNA <Jump If Not Above>	Lompat, jika Operand1 < Operand2 dengan CF=1 atau ZF=1
JNAE <Jump If Not Above nor Equal>	Lompat, jika Operand1 < Operand2 dengan CX=1
JNB <Jump If Not Below>	Lompat, jika Operand1 > Operand2 dengan CF=0
JNBE <Jump If Not Below nor Equal>	Lompat, jika Operand1 > Operand2 dengan CF=0 dan ZF=0
JNC <Jump If No Carry>	Lompat, jika CF=0
JNG <Jump If Not Greater>	Lompat, jika Operand1 <= Operand2 dengan ZF=1 atau SF tidak sama OF
JNGE <Jump If Not Greater Nor Equal>	Lompat, jika Operand1 <= Operand2 dengan SF tidak sama OF
JNL <Jump If Not Less>	Lompat, jika Operand1 >= Operand2 dengan SF=OF
JNLE <Jump If Not Less Nor Equal>	Lompat, jika Operand1 > Operand2 dengan ZF=0 dan SF=OF
JNO <Jump If No Overflow>	Lompat, jika tidak terjadi tidak terjadi Overflow
JNP <Jump If Not Parity>	Lompat, jika Ganjil
JNS <Jump If No Sign>	Lompat, jika SF=0
JNZ <Jump If Not Zero>	Lompat, jika tidak 0
JO <Jump On Overflow>	Lompat, jika OF=1
JP <Jump On Parity>	Lompat, jika Genap
JPE <Jump If Parity Even>	Lompat, jika PF=1
JPO <Jump If Parity Odd>	Lompat, jika PF=0
JS <Jump On Sign>	Lompat, jika SF=1
JZ <Jump Is zero>	Lompat, jika 0

Tugas Pendahuluan

1. Buatlah Program dengan memanfaatkan *flow control Instruction* sebagai berikut :

```
masukkan 2 huruf      : ZM
sorting Huruf         : MZ
```

2. Buatlah Program sebagai berikut :

```
masukkan sebuah kalimat  : KOMPUTER
Jumlah Konsonan : 5
```

3. Buatlah Program dengan tampilan sebagai berikut :

```
inputkan (Angka 1-9) : 5
Output
*
* *
* * *
* * * *
* * * * *
```

Latihan

1. Translasi bahasa tingkat tinggi ke bahasa assembly

Bahasa Tingkat Tinggi	Bahasa Assembly
IF AX < 0 THEN replace ax by -ax END_IF	; IF AX < 0 CMP AX,0 ;AX < 0 ? JNL END_IF ; no, exit ; THEN NEG AX END_IF

FOR 80 kali DO
Tampilkan '*'
END_FOR
	MOV CX,80
	MOV AH,2
	MOV DL, '*'
	TOP:
	INT 21h
	LOOP TOP

cl = 1
while (cl <=80)
tampilkan '*'
inc cl
end while
	MOV CL, 0
	WHILE_ : CMP CL, 80
	JA END_WHILE
	MOV AH, 02H
	MOV DL, '*'
	INT 21H
	INC CL
	JMP WHILE_

	END_WHILE: RET

```
2.   ORG 100H
     MULAI:
             MOV BX,0
             MOV CX, 0

     TAMBAH:
             CMP CX, 10
             JA EXIT
             ADD BX, CX
             ADD CX, 2
             JMP TAMBAH
     EXIT:
             RET
3.   ORG 100H
     MULAI:
            MOV AH, 09H
            LEA DX, KAL1
            INT 21H

            MOV AH, 01H
            INT 21H

            MOV BL,AL
            CMP BL, 'x'
            JE EXIT
            JMP MULAI
     EXIT:
            MOV AH, 09H
            LEA DX, KAL2
            INT 21H
            RET
     KAL1 DB 13,10, 'INPUT KARAKTER : $'
     KAL2 DB 13,10, 'PROGRAM SELESAI $'
```

Tugas Praktikum

1. Translasikan bahasa tingkat tinggi dibawah ini ke bahasa assembly :

   a. CX ← O

      WHILE (AX > = BX)

         CX ← CX - 1

         AX = AX - BX

END WHILE

b. CX ← 0

REPEAT

AX = AX + CX

BX = BX - 1

UNTIL (BX = 0)

2. Buatlah program untuk menjumlahkan data ke dalam register AX

a. 1 + 4 + 7 + … + 148

b. 100 + 95 + 90 + . . . +5

3. Buat program untuk menginputkan beberapa karakter (menggunakan int 21h service 01h), kemudian lakukan seleksi apakah terdapat huruf vokal.

4. Buat Program dengan tampilan sebagai berikut :

```
Input angka (1-9) : 5
Output
5 5 5 5 5
5 5 5 5
5 5 5
5 5
5
```