PRAKTIKUM MIKROPROSESOR DAN BAHASA RAKITAN

FUNGSI PENGAMBIL KEPUTUSAN

TUJUAN

1. Mahasiswa memahami dan mengerti tentang berbagai operasi pengambil keputusan dalam bahasa mesin

DASAR TEORI

Sama seperti pada pemrograman level tinggi, maka pemrograman bahasa mesin juga mengenal perintah-perintah untuk penentuan keputusan. Fungsi ini dilakukan menggunakan instruksi *jump* (JMP) dan *compare* (CMP). *Jump* (JMP) merupakan perintah untuk mengarahkan eksekusi suatu program ke suatu alamat tertentu. Instruksi ini bisa membutuhkan suatu syarat maupun tidak tergantung dari kebutuhan programnya. Sintaknya sebagai berikut :

JMP Tujuan

Perintah JMP yang menggunakan syarat pada umumnya menggunakan instruksi bit dan logika maupun instruksi pembandingan sebagai awal dari lompatannya. Contohnya TEST atau CMP yang diikuti dengan JNZ, JNE, ataupun instruksi lompat bersyarat lainnya.

Perintah perbandingan dengan CMP akan membandingkan 2 buah operan dengan sintak sebagai berikut :

CMP Operan1, Operan2

Operasinya yaitu operasi pengurangan antara Operan1 dan Operan2. Sama seperti TEST, maka instruksi ini hanya akan mengubah *flag register* saja, bukan operan-operannya. Adapun *flag* yang bisa berubah antara lain :

Flag	Kondisi
OF = 1	Operan1 < Operan2 untuk operasi bilangan bertanda
SF = 1	Operan1 < Operan2 untuk operasi bilangan bertanda
ZF = 1	Operan1 = Operan2
CF = 1	Operan1 < Operan2 untuk operasi bilangan tak bertanda

Instruksi CMP biasanya akan diikuti dengan instruksi lompat bersyarat dengan kondisi berbeda-beda sesuai tabel berikut ini:

Instruksi	Kondisi
JA (Jump if above)	Operan1 > Operan2 (bilangan tak bertanda)
JG (Jump if greater)	Operan1 > Operan2 (bilangan bertanda)
JE (Jump if equal)	Operan1 = Operan2

JNE (Jump if not equal)	Operan1 <> Operan2
JB (Jump if below)	Operan1 < Operan2 (bilangan tak bertanda)
JL (Jump if less)	Operan1 < Operan2 (bilangan bertanda)
JBE (Jump if below or equal)	Operan1 <= Operan2 (bilangan tak bertanda)
JLE (Jump if less or equal)	Operan1 <= Operan2 (bilangan bertanda)
JAE (Jump if above or equal)	Operan1 >= Operan2 (bilangan tak bertanda)
JGE (Jump if greater or equal)	Operan1 >= Operan2 (bilangan bertanda)

Sedangkan untuk instruksi yang termasuk dalam lompat bersyarat di mana jarak tujuan harus dalam jangkauan -128 sampai dengan +127 byte dari tempat melompatnya, selain instruksi pada tabel sebelumnya, juga terdapat instruksi lainnya seperti dapat dilihat pada tabel berikut :

Instruksi	Kondisi
JC (Jump if carry)	CF = 1
JCXZ (Jump if CX is zero)	CX = 0
JNA (Jump if not above)	Operan1 < Operan2 dengan CF atau ZF = 1
JNAE (Jump if not above or equal)	Operan1 < Operan2 dengan CX =1
JNB (Jump if not below)	Operan1 > Operan2 dengan CF = 0
JNBE (Jump if not below nor equal)	Operan1 > Operan2 dengan CF atau ZF = 0
JNC (Jump if no carry)	CF = 0
JNG (Jump if not greater)	Operan1 < Operan2 dengan ZF = 1 atau SF <> OF
JNGE (Jump if not greater nor equal)	Operan1 <= Operan2 dengan SF <> OF
JNL (Jump if not less)	Operan1 >= Operan2 dengan SF = OF
JNLE (Jump if not equal nor less)	Operan1 > Operan2 dengan ZF = 0 dan SF = OF
JNO (Jump if no overflow)	OF = 0
JNP (Jump if not parity)	Hasil operasi ganjil
JNS (Jump if no sign)	SF = 0
JNZ (Jump if not zero)	Hasil operasi <> 0
JO (Jump on overflow)	OF = 1
JP (Jump on parity)	Hasil operasi genap
JPE (Jump on parity even)	PF = 1
JPO (Jump if parity odd)	PF = 0
JS (Jump on sign)	SF = 1
JZ (Jump on zero)	Hasil operasi = 0

Stack merupakan bagian dari memori yang dipergunakan untuk penyimpanan data sementara. Instruksinya meliputi PUSH, POP, PUSF, POPF. Pada program satu segmen maka akan digunakan pasangan register SS:SP sebagai penunjuk lokasi stack. Stack akan melakukan penyimpanan suatu data dan pengeluaran kembali datanya dengan sistem LIFO (Last In First Out). Untuk memasukkan data ke stack maka dipergunakan PUSH, dan untuk mengeluarkan lagi datanya dipergunakan POP. Urutan PUSH dan POP harus mengikuti sistem LIFO. PUSHF dan

POPF memiliki fungsi yang sama dengan PUSH dan POP tapi untuk datanya adalah flag register. Jadi PUSHF sama saja dengan PUSH flag register, dan POPF sama saja dengan POP flag register.

PROSEDUR PERCOBAAN

A. Instruksi lompat

1. Tuliskan program *assembly* berikut menggunakan TASM , lalu *lingking* menjadi COM :

```
.Model Small
.Code
ORG 100h
TData:
   JMP Proses
   BilA db 67
   BilB db 66
   Kal0 db 'Bilangan A lebih kecil dari bilangan B $'
   Kal1 db 'Bilangan A sama dengan bilangan B $'
   Kal2 db 'Bilangan A lebih besar dari bilangan B $'

Proses:
   MOV AL,BilA ; Masukkan bilangan A pada AL
   CMP AL,BilB ; Bandingkan AL(BilA) dengan Bilangan B
   JB AKecil ; Jika BilA < BilB, lompat ke AKecil
   JE Sama ; Jika BilA = BilB, lompat ke Sama
   JA ABesar ; Jika BilA > BilB, lompat ke ABesar

Akecil:
   LEA DX,Kal0 ; Ambil offset Kal0
   JMP Cetak ; Lompat ke cetak

Sama:
   LEA DX,Kal1 ; Ambil offset Kal1
   JMP Cetak ; Lompat ke cetak

ABesar:
   LEA DX,Kal2 ; Ambil offset Kal2

Cetak:
   MOV AH,09 ; Servis untuk mencetak kalimat
   INT 21h ; Cetak kalimat !!

EXIT:
   INT 20h ; Kembali ke DOS.
End      TData
```

2. Jalankan programnya dan lihat hasilnya. Ubahlah programnya sehingga ketiga kondisinya dapat terjadi.

B. Lompat bersyarat

1. Tuliskan program *assembly* berikut menggunakan TASM , lalu *lingking* menjadi COM :

```
.Model Small
.Code
ORG 100h
TData :
   JMP Proses
   Kal db ' Lucky Luck menembak ',13,10
   db 'Lebih cepat dari bayangannya !! ',7,7,'*'

Proses:
   XOR BX,BX ; BX=0
   MOV AH,02h ; Servis Untuk Cetak Karakter

Ulang:
   CMP Kal[BX],'*' ; Bandingkan dengan '*'
   JE Exit ; Jika Sama Lompat ke Exit
   MOV DL,Kal[BX] ; Masukkan karakter ke BX menuju DL
   INT 21h ; Cetak karakter
   INC BX ; Tambah 1 pada BX
   JMP Ulang ; Lompat Ke Ulang

Exit :
   INT 20h ; Selesai ! kembali ke DOS
End TData
```

2. Jalankan programnya dan amati yang terjadi. Ubah JE Exit dengan instruksi lompat lain yang dapat menghasilkan output yang sama.

C. Stack software

1. Tuliskan program *assembly* berikut menggunakan TASM , lalu *lingking* menjadi COM :

```
.Model Small
.Code
ORG 100h
TData :
   JMP Proses
   Kal db 'LANG LING LUNG $'
   Ganti db 13,10,'$'
   Stacks dw ?

Proses:
   LEA DX,Kal
   MOV Stacks,DX
   MOV AH,09
   INT 21h
   LEA DX,Ganti
   INT 21h
   MOV DX,Stacks
   INT 21h

Exit :
```

```
   INT 20h
End TData
```

3. Jalankan programnya dan amati yang terjadi. Mengapa program ini disebut menggunakan stack software?

D. Stack hardware

1. Tuliskan program *assembly* berikut menggunakan TASM , lalu *lingking* menjadi COM :

```
.Model Small
.Code
ORG 100h
TData :
   JMP Proses
   Kal db 'LANG LING LUNG $'
   Ganti db 13,10,'$'
   Stacks dw ?

Proses:
   LEA DX,Kal
   PUSH DX
   MOV AH,09
   INT 21h
   LEA DX,Ganti
   INT 21h
   POP DX
   INT 21h
Exit :
   INT 20h
End TData
```

2. Jalankan programnya dan amati yang terjadi. Mengapa program ini disebut menggunakan stack hardware?

TUGAS

1. Sebutkan 5 macam instruksi lompat bersyarat dengan kondisi melompatnya?
2. Buatlah program untuk mengecek apakah masukan dari keyboard sebuah angka atau huruf kecil. Sertakan screenshot dari hasil program Anda.