PERCOBAAN 4
INSTRUKSI PERCABANGAN

Oleh : Sumarna, Jurdik Fisika, FMIPA, UNY
E-mail : **sumarna@uny.ac.id**

Tujuan dari percobaan ini adalah untuk membiasakan diri dengan instruksi percabangan baik yang bersyarat maupun tak bersyarat, mempelajari tehnik merancang program dengan iterasi (loop), dan melatih menggunakan suatu register status untuk keperluan pengambilan keputusan.

Mikrokomputer (khususnya mikroprosesor) melaksanakan program dengan cara langkah demi langkah dalam urutan tertentu, dimulai alamat pertama dari suatu program (misalkan dimulai dari alamat 1800H). Urutan alamat suatu program dapat berubah bila dijumpai instruksi lompat (JUMP) yaitu suatu instruksi percabangan. Pelaksanaan program dimulai ketika ada perintah START atau RUN atau GO. Perintah tersebut akan menginisialisasi Program Counter (PC) untuk memulai penghitungan/cacahan pada nol atau keadaan awal tertentu. **PC adalah semacam pencacah di dalam Unit Kendali yang memuat alur alamat program yang akan dilaksanakan mikrokomputer secara urut**. PC di dalam MPF-I merupakan register 16 bit. Jika keadaan pin RESET (pada mikroprosesor) dari 0 menjadi 1, maka register PC akan di-set pada keadaan 0000H. Pelaksanaan program kemudian dimulai dari alamat 0000H menurut pulsa detak yang dihasilkan oleh CLOCK. Setiap kalai CPU selesai mengambil 1 byte untuk setiap instruksi dari memori, secara otomatis PC akan bertambah dengan 1. Setelah suatu program dilaksanakan sampai alamat tertentu, PC dapat diubah ke alamat yang lain bilamana pemrogram tidak menghendaki pelaksanaan pada alamat berikutnya. Program kemudian meloncat ke alamat yang lain dan melanjutkan pelaksanaam program mulai alamat itu. Misalnya mnemonic LD PC,1828H berarti PC akan diubah ke alamat 1828H maka pelaksanaan program berikutnya mulai dari alamat 1828H. Pada MPF-I Z80 perintah untuk memaksa PC ke alamat 1828H itu adalah perintah lompat (JP : Jump) yang digunakan untuk menyatakan adanya perubahan urutan dalam pelaksanaan program. Jadi LD PC,1828H sama dengan perintah JP 1828H.

Instruksi percabangan bersyarat melakukan operasi lompat bila beberapa syarat tertentu terpenuhi. Syarat-syarat ini tergantung dari data pada register Flag. Fungsi tersebut menyebabkan mikrokomputer mampu menanggapi berbagai syarat yang datangnya dari luar. Hal tersebut juga merupakan alat dalam perancangan program iterasi (loop) atau percabangan. Berikut ini mnemonic untuk instruksi percabangan bersyarat dalam MPF-I Z80 :

JP C,xxxx Jika ada carry atau carry flag = 1, lompat ke alamat xxxx.

JP NC,xxxx Jika tidak ada carry atau carry flag = 0, lompat ke alamat xxxx.

JP Z,xxxx Jika zero flag = 1, atau hasil dari operasi yang terdahulu adalah 0, maka lompat ke alamat xxxx.

JP NZ,xxxx Jika zero flag = 0, lompat ke alamat xxxx.

JP PE,xxxx Jika parity/overflow flag = 1 (genap), atau jika terjadi overflow dalam operasi aritmatik sebelumnya, maka lompat ke alamat xxxx.

JP PO,xxxx Jika parity/overflow flag = 0 (ganjil), atau tidak ada overflow, maka lompat ke alamat xxxx.

JP P,xxxx Jika sign flag = 0 (tanda dari hasil operasi yang terdahulu adalah positif), maka lompat ke alamat xxxx.

JP M,xxxx Jika sign flag = 1 (negatif), maka lompat ke alamat xxxx.

Untuk mengurangi memori yang ditempati program (menekan biaya sistem mikrokomputer), maka pada MPF-I Z80 dapat memakai address relatif untuk menunjukkan lompatan suatu program. Untuk setiap satu operasi lompat dapat menghemat memori 1 byte dibandingkan bila menggunakan instruksi JP yang membutuhkan 2 byte address absolut. Beberapa instruksi Jump Relative (JR) adalah :

JR 10H Lompat ke depan 10H (= 16) lokasi dari PC yang saat itu dipakai (address instruksi berikutnya). Sebenarnya, address instruksi yang akan dilaksanakan berikutnya didapat dengan menambah 10H pada PC yang sedang dipakai.

JR C,F0H Jika carry flag = 1, lompat ke belakang 10H lokasi. Karena bit F0H yang paling kiri adalah 1, hal ini diketahui sebagai bilangan negatif (komplemen 2 –nya adalah 10H).

JR NC,7FH Jika carry flag = 0, lompat ke depan 127 lokasi (nilai maksimum).

JR Z,80H Jika zero flag = 1, yaitu bila hasil dari operasi yang terdahulu adalah 0, lompat ke belakan 128 lokasi. 80H (= -128) adalah bilangan negatif minimum yang dapat digunakan dalam address relative.

Terlihat bahwa address relative yang positif berarti lompat ke depan, dengan lompatan yang paling besar adalah +127 (7F). Address relative negatif berarti lompat ke belakang, dengan lompatan terbesar ke belakang adalah –128 (80H). Lompatan ini diukur dari alamat mnemonic (Op-Code) instruksi berikutnya. Lompatan bersyarat bergantung pada status dari carry flag atau zero flag. Pada MPF-I Z80, sign flag atau parity/overflow flag tidak dapat digunakan sebagai syarat jump relative.

Keunggulan dari mikrokomputer yang paling penting adalah dapat mengulangi langkah-langkah yang diperlukan dalam menyelesaikan suatu tugas sebanyak mungkin sampai tugas yang bersangkutan selesai dilaksanakan. Fungsi ini dikerjakan dengan program loop atau iterasi. Program loop dasar harus memuat :

1. Suatu preset (nilai awal) pencacah loop yang menyatakan cacah loop yang harus dilakukan. Biasanya suatu register general dalam CPU dapat dipakai sebagai pencacah loop.
2. Pencacah loop dikurangi dengan 1 setiap kali suatu putaran loop selesai dilaksanakan. Setiap selesai suatu putaran, nilai pencacah loop harus diperiksa. Jika pencacah loop tidak sama dengan 0, loop diulangi sampai pencacah loop sama dengan 0.

Percobaan 4.1 :

Susunlah suatu program yang dapat digunakan untuk menjumlahkan data 8 bit pada alamat memori dari 1900H s/d 190FH dan hasilnya disimpan dalam pasangan register DE. Berikut ini mnemonic sebagai petunjuk dalam pembuatan program yang diinginkan.

Label	Mnemonic	Keterangan
	LD C,10H	Register C sebagai pencacah loop. Karena 16 (10H) data 8 bit akan dijumlahkan bersama, maka nilai awal 10H di-set di regiater C. (1800 0E 10)
	XOR A	Nol-kan accumulator dan carry flag. (1802 AF)
	LD HL,1900H	Register HL digunakan sebagai penunjuk address. Isi memeori yang address-nya ditunjukkan oleh HL akan ditambahkan ke register A. Address pertama adalah 1900H. (1803 21 00 19)
	LD D,A	Register D digunakan untuk menyimpan Carry yang dihasilkan pada operasi penjumlahan. Nol-kan register D. (1806 57)
XX	ADD A,(HL)	Tambahkan isi memori yang address-nya ditunjukkan oleh HL ke register A. Instruksi ini akan diulang sebanyak 16 kali. (1807 86)
	INC HL	Menambah HL dengan 1. HL yang baru menunjuk pada data berikutnya yang ada pada memori untuk ditambahkan pada register A. (1808 23)
	JR NC,YY	Jika tidak dihasilkan carry, lompat ke address YY untuk melanjutkan pelaksanaan program. (1809 30 01)
	INC D	Jika dihasilkan carry, tambahkan carry ini pada register D. (180B 14)
YY	DEC C	Kurangi register C dengan 1. (180C 0D)
	JR NZ,XX	Jika hasilnya tidak sama dengan 0 (zero flag = 0), program loop belum selesai. Lompat ke XX untuk mengulangi loop. (180D 20 F8)
	LD E,A	Jika zero flag = 1, semua data telah ditambahkan bersama-sama. Masukkan A pada E, hasilnya akan tersimpan dalam pasangan register DE. (180F 5F)
	END	Selesai. (1810 76) atau (1810 FF)

Catatan : Sebelum menjalankan program, masukkan data sembarang ke dalam setiap lokasi memori dari 1900H s/d 190FH (sejumlah 16 data masing-masing 8 bit).

Masukkan program yang telah tersusun ke dalam MPF-I Z80, kemudian jalankan, dan periksalah isi dari pasangan register DE. Cocokan hasilnya dengan perhitungan secara manual.

Percobaan 4.2 :

Jika cacah loop kurang dari 256, register B dianggap sebagai penghitung loop. Pada akhir loop, perintah DJNZ dapat dipakai untuk mengurangi isi register B dengan 1 (satu). Jika hasilnya tidak sama dengan 0 (nol), loncat ke lokasi yang ditunjuk dengan menggunakan metode jump relative untuk melanjutkan pelaksanaan program. Pelajari program berikut, apa fungsinya, dan cobalah dengan memasukkannya ke dalam MPF-I.

```
        ORG   1800h
        LD    HL,1900h
        LD    B,20h
LOOP :  LD    (HL),A
        INC   HL
        DJNZ  LOOP
        RST   38h
```

1. Pre-set register A pada 0 (nol), kemudian jalankan program di atas. Periksalah isi memori yang alamat lokasinya dari 1900h s/d 191Fh. Periksa juga isi memori yang alamat lokasinya di atas 191Fh (5 lokasi saja).
2. Pre-set register A pada 55, kemudian jalankan program di atas. Periksalah isi memori yang alamat lokasinya dari 1900h s/d 191Fh.
3. Pre-set register A pada 64, gantilah perintah **LD B,20h** dengan perintah **LD B,0h** dan kemudian jalankan program di atas. Periksalah isi memori yang alamat lokasinya dari 1900h s/d 191Fh.
4. Diskusikan masalah di atas !

Percobaan 4.3 :

Dalam program yang lebih rumit, loop dapat bersarang di dalam loop yang lain secara total. Selidiki program berikut, apa fungsinya, dan cobalah dengan memasukkannya ke dalam MPF-I.

```
        ORG   1800h
        LD    HL,19FFh
        LD    C,0Fh
LOOP2 : LD    B,10h
```

```
LOOP1 :   LD    (HL),C
          DEC   HL
          DJNZ  LOOP1
          DEC   C
          JPNZ  LOOP2
          RST   38h
```

Periksalah isi memori yang alamatnya :

a. 1910h s/d 191Fh
b. 1920h s/d 192Fh
c. 1930h s/d 193Fh
d. …

   dst.

o. 19F0h s/d 19FFh

Percobaan 4.4 :

Jika cacah loop lebih dari 256, suatu register 16 bit dapat dipakai sebagai penghitung loop. Tetapi penambahan atau pengurangan isi suatu register 16 bit tidak mempengaruhi status flag. Sehingga perlu perintah-perintah pembantu untuk menentukan apakah penghitung loop telah sama dengan 0 (nol). Program berikut men-set isi memori dengan alamat lokasi 1880h s/d 19FFh dengan data AAh. Pelajarilah dengan seksama ! Kapan proses looping berhenti ? Periksalah status flagnya !

```
          ORG   1800h
          LD    C,01h
          LD    B,80h
          LD    HL,1880h
LOOP :    LD    (HL),0AAh
          INC   HL
          DJNZ  LOOP
          DEC   C
          JRNZ  LOOP
          RST   38h
```

Percobaan 4.4 :

Suatu program loop bisa tidak menggunakan penghitung mundur (ke bawah). Fungsi penghitung mundur dapat diganti dengan penghitung maju (ke atas) atau dengan menggunakan metode perbandingan alamat ataiu metode perbandingan data. Pelajari metode yang dipakai dalam program loop berikut !

1. Memindahkan data pada daerah memori dengan alamat awal 1B00h ke daerah memori dengan alamat awal 1A00h. Perpindahan akan berkenti ketika menemukan data 0FFh.

```
        ORG   1800h
        LD    HL,1B00h
        LD    DE,1A00h
LOOP :  LD    A,(HL)
        LD    (DE),A
        CP    0FFh
        JRZ   EXIT
        INC   DE
        INC   HL
        JR    LOOP
EXIT :  RST   38h
```

2. Gantilah semua data yang tersimpan di daerah memori yang dimulai pada alamat yang ditunjukkan oleh HL sampai dengan alamat yang ditunjukkan oleh DE dengan nilai komplemen-2 nya. Nilai HL dan DE harus di-set lebih dahulu. Nilai HL harus lebih besar dari pada nilai DE.

```
        ORG   1800h
LOOP :  LD    A,(HL)
        NEG
        LD    (HL),A
        INC   HL
        AND   A
        SBC   HL,DE
        ADD   HL,DE
        JRNZ  LOOP
        RST   38h
```