BAHASA PEMOGRAMAN MIKROPROSESOR Z80

Yoyo somantri
Dosen Jurusan Pendidikan Teknik Elektro
FPTK Universitas Pendidikan Indonesia

Pendahuluan

Pada bab ini akan dibahas tujuan perkuliahan, bahasa pemrograman mikroprosesor Z80. Selain itu dikemukakan contoh-contoh bahasa program sederhana dan aplikasinya.

Tujuan Perkuliahan

Setelah mempelajari bab ini, diharapkan mahasiswa mampu untuk:

1. Memahami bahasa pemograman pada mikroprosesor Z80 .
2. Mengaplikasikan set instruksi dalam menyelesaikan suatu masalah.
3. Membuat program-program aplikasi dengan menggunakan bahasa *assembly*.

1. Pemograman Z 80

Di dalam pemograman terdapat beberapa level, ada yang disebut level tinggi seperti : Bahasa Pascal, Visual Basic, paket program dll. Sedangkan level taraf rendaha biasanya seperti : Bahasa mesin, dan bahasa assembly. Program adalah susunan instruksi yang logis dan mengandung bahasa yang diketahui oleh mikroprosesor dan bila dieksekusi akan diperoleh suatu hasil yang sesuai dengan instruksi pada program. Sebelum suatu program dilaksanakan oleh CPU, program harus disimpan dahulu di dalam memori dalam bentuk bilangan biner. Program jenis ini disebut program dalam bahasa mesin (*machine language program*). Hanya bahasa mesin yang dapat dimengerti oleh sebuah sistem mikroprosesor. Penulisan dalam bahasa mesin ini biasanya dinyatakan dalam bilangan heksadesimal. Misalnya instruksi 8 bit 1011 1110B (B menyatakan biner) bila dinyatakan dalam bilangan hexadesimal menjadi BEH (H menyatakan heksadesimal). Bagi seorang pemakai sistem mikroprosesor, menginterpretasikan program dalam bahasa mesin sangatlah sulit dan membutuhkan banyak waktu. Para pembuat mikroprosesor telah membagi instruksi-instruksi tersebut menjadi beberapa kategori menurut fungsinya. Instruksi – instruksi CPU dan *register* biasanya dinyatakan dalam simbol yang disebut mnemonic.

Mnemonic adalah kumpulan instruksi-instruksi. Misalnya, LD A, L (masukan data dari *register* L ke *register* A). Program yang ditulis dalam kode *mnemonic* disebut program dalam bahasa *assembly*. Sebelum suatu program dalam bahasa *assembly* dilaksanakan oleh CPU, program tersebut harus diterjemahkan dalam bahasa mesin oleh program khusus yang disebut *assembler*.

Keunggulan dari program dalam bahasa *assembly* terhadap program dalam bahasa mesin adalah program dalam bahasa *assembly* jauh lebih cepat membuatnya, *mnemonic – mnemonic*-nya membuat para pemakai lebih mudah mengingat instruksi setnya dan biasanya *assembler* telah mempunyai paket *self diagnostic* untuk memeriksa program yang dibuat apabila ada kesalahan. Kekurangan utama program dalam bahasa *assembly* adalah membutuhkan sebuah *assembler* (penterjemah ke bahasa mesin).

Sumber dari Guru Mikro Saya. Inelco Bandung (1986).

2. Analisa Masalah

Untuk membuat suatu program terlebih dahulu kita harus membuat analisa yang terperinci dari masalah yang akan dibuatkan programnya. Hal – hal yang harus diperhatikan adalah :

1. Karakteristik dan tuntutan masalah
2. Kondisi – kondisi yang telah diketahui
3. Format informasi input dan bagaimana format itu dikonversikan
4. Format data output dan bagaimana format itu dikonversikan
5. Jenis data dan ketelitiannya
6. Waktu pelaksanaan program yang dibutuhkan
7. Instruksi – instruksi CPU dan sifat-sifatnya
8. Besarnya memori
9. Kemungkinan dapat/ tidaknya masalah tersebut diselesaikan
10. Metoda pemecahan masalah
11. Evaluasi program
12. Bagaimana/dimana hasil pembuatan program akan disimpan

Diagram alir (flowchart) urutan ini dapat dilihat pada gambar.

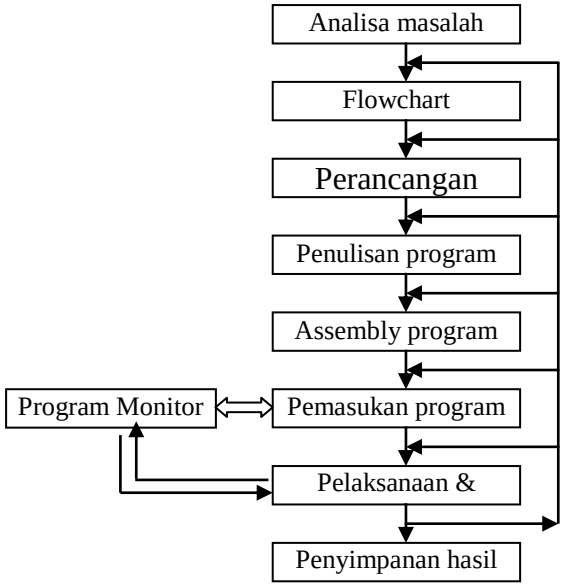
3. *Flowchart*

Flowchart atau diagram alir adalah suatu skema yang menggambarkan urutan kegiatan dari suatu program dari awal hingga akhir. Bila suatu *flowchart* lengkap telah selesai dibuat, gambaran lengkap tentang proses pemikiran seorang *programmer* dalam memecahkan suatu masalah dapat diikuti. *Flowchart* juga sangat penting untuk pemeriksaan program yang telah selesai, juga dapat membantu orang lain dalam memahami algoritma yang tepat yang dibuat oleh *programmer*. *Flowchart* dapat dibagi 2, yaitu :

1. *Flowchart* sistem : menunjukkan jalannya program secara umum (garis besarnya saja).

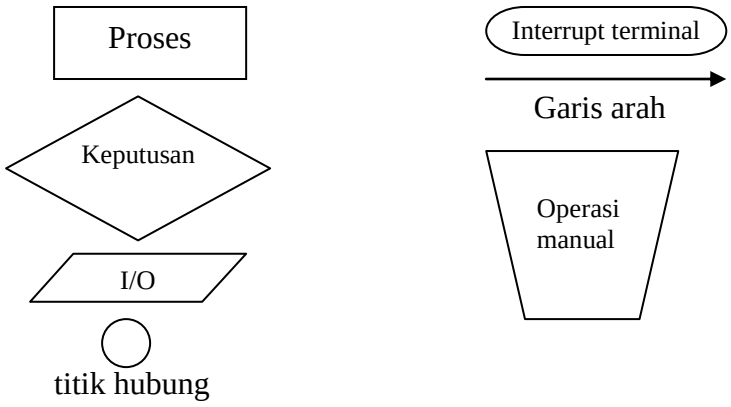
2. *Flowchart* terperinci : berisi perincian – perincian (detail) yang penting untuk programmer.

Biasanya suatu program yang rumit didahului dengan *flowchart* sistem lalu dilengkapi dengan *flowchart* terperinci. Keuntungan dari sebuah *flowchart* adalah urutan langkah – langkah dari sebuah program ditunjukkan dengan simbol anak panah juga digunakan simbol-simbol lain untuk menunjukkan operasi yang dilaksanakan pada tiap-tiap langkah dari program tersebut. Simbol-simbol standar yang digunakan dalam pembuatan sebuah flowchart dapat dilihat pada gambar 2



Gambar 1. Analisa masalah

Sumber dari Guru Mikro Saya. Inelco Bandung (1986).



Gambar 2. Simbol – simbol standar flowchart

4. Merancang Program

Program ada beberapa jenis. Program – program yang sederhana, misalnya program persamaan matematika, konversi sinyal input dan output, program untuk menkode dan mendekode data, program untuk menjalankan peripheral. Program – program yang lebih rumit, misalnya program untuk *assembler*, *monitor*, *control system* atau aplikasi peralatan – peralatan khusus. Hal – hal yang harus diperhatikan dalam merancang suatu program adalah :

1. Mendapatkan sinyal input atau data
2. Menghasilkan atau mengkonversi sinyal output dan data
3. Perhitungan dan analisa logika di dalam program utama
4. Hubungan antara program utama dan *sub routine*
5. Penggunaan dari *register-register* yang ada di dalam CPU
6. Alokasi penggunaan memori pada program utama
7. Alokasi penggunaan memori pada *subroutine*
8. Alokasi memori untuk tabel data dan metoda *index addressing*
9. Program inisialisasi dan konstanta-konstanta
10. Definisi variabel-variabel di dalam program
11. Pertimbangan dari urutan waktu dan kecepatan pelaksaan program
12. Keterbatasan kapasitas memori
13. Panjang dan kepresisian data

Sumber dari Guru Mikro Saya. Inelco Bandung (1986).

5. Penulisan Program

Pada umumnya program pada mikroprosesor ditulis dalam bahasa *assembly* dan bahasa mesin. Suatu statemen dalam program terdiri dari 4 bagian yaitu label, *op-code*, *operand* dan keterangan. Label berisi suatu nama/tanda bukan alamat absolut. Penggunaan label memudahkan untuk mereferensi instruksi atau untuk menetapkan operasi percabangan. *Op-code* (kode operasi) berisi singkatan-singkatan dan instruksi yang akan dilaksanakan, misalnya INC untuk *increment* (operasi penjumlahan). Operand terdiri dari sintaks yang berubah-ubah sesuai dengan instruksi yang harus dilaksanakan. Pada penulisan program dalam bahasa *assembly* yang sulit adalah penulisan *operand*-nya. Pemberian keterangan berfungsi untuk menjelaskan operasi pelaksanaan dari suatu instruksi. Suatu *statement* program tanpa keterangan kadang-kadang sulit dimengerti. Keterangan tersebut lebih dirasakan artinya pada program-program yang rumit.

6. Program *Assembly*

Cara yang paling efektif untuk menterjemahkan program bahasa *assembly* ke bahasa mesin adalah dengan mempergunakan *assembler* (penterjemah) yang telah tersedia dalam system mikrokomputer tersebut. Tetapi bagi seorang pemula atau programmer yang belum mengenal sistem pengembangan mikrokomputer tersebut dapat pula menterjemahkan program secara manual. Prosedurnya adalah sebagai berikut :

1. Terjemahkan setiap instruksi (*mnemonic*) kedalam kode mesin berdasarkan tabel konversi.
2. Setelah menentukan alamat dari awal program (yang berada pada alamat RAM pemakai), tentukan alamat yang sesuai untuk byte pertama masing-masing instruksi. Jumlah byte yang dibutuhkan harus disediakan secara tepat termasuk juga byte-byte yang akan diisi kemudian, misalnya dalam instruksi JP, DJNZ serta *address* tujuan pada instruksi JP, CALL dan lain-lain.
3. Menghitung percabangan relatif dan mengisikannya ke dalam program yang telah ditulis dalam bahasa mesin. Rumus sederhana untuk menghitung percabangan relatif (pergeseran) :

   Pergeseran = (alamat tujuan) – (alamat instruksi berikutnya).

Jika hasil perhitungan adalah positif, maka hasil perhitungan tersebut adalah nilai yang kita isikan. Jika hasil perhitungan adalah negatif, maka tambahkan 100H pada hasil tersebut (akan diperoleh nilai komplemen keduanya) dan hasil akhirnya merupakan hasil yang dapat kita isikan. Atau bila hasilnya negatif, maka dikomplemenkan dan ditambah satu. Misalnya pada instruksi DJNZ label, *address* tujuannya berada pada 0002H, address instruksi berikutnya (*address* setelah instruksi DJNZ) adalah 1016H, instruksi DJNZ kode bahasa mesinnya adalah 10xx (xx adalah pergeseran yang harus dihitung).

xx (pergesaran) = 0002H (*address* tujuan) – 1016H (*address* instruksi selanjutnya) = - 14H

xx (pergeseran) = 100H + (-14H) = 100 – 14H = ECH, sehingga instruksi DJNZ label tersebut bila diterjemahkan ke dalam kode mesinnya adalah 10ECH.

Sumber dari Guru Mikro Saya. Inelco Bandung (1986).

7. Pengisian Program

Untuk pengisian program terlebih dahulu kita harus mengetahui peta memori dari sistem yang akan kita gunakan. Program yang kita isi tersebut harus dimasukkan pada address memori untuk RAM pemakai. Setelah program diisi ke dalam RAM diperlukan proses pemeriksaan kesalahan untuk menghilangkan kesalahan-kesalahan yang mungkin

terjadi. Instruksi atau data yang tertinggal dapat diselipkan pada *address* yang diiinginkan dengan mengisikan kembali program tersebut atau dengan menggunakan *block data transfer*. Bila kita merevisi program, kita perlu memeriksa apakah instruksi-instruksi loncat (*jump*) yaitu JP, JR, DJNZ, CALL dan sebagainya terpengaruh oleh perubahan *address-address* pada memori, jika hal ini terjadi, kita harus segera memperbaikinya.

Sumber dari Guru Mikro Saya. Inelco Bandung (1986).

8. Menjalankan dan Memeriksa Program

Sebelum menjalankan suatu program, kita harus mengeset parameter-parameter inisialisasi dan meletakkan penghitung program (*program counter*) pada *address* awal program. Dengan menekan tombol yang berfungsi untuk menjalankan program, pelaksanaan program akan dimulai. Setelah program selesai dilaksanakan, periksalah hasilnya. Jika ada kesalahan, program harus diperiksa langkah demi langkah dengan bantuan *program monitor*. Setelah program selesai diperbaiki, jalankan sekali lagi dan periksa hasilnya.

9. Contoh – contoh program :

Percobaan Transfer Data

1. Program bahasa *assembly* ditulis untuk menyusun isi *register* berikut :
   A = 0, B = 1, C = 2, D = 3, E = 4, H = 5, L = 6 (gunakan instruksi LD 8 bit untuk mentransfer 1 byte data).

Alamat	Bahasa Mesin	Bahasa *Assembly*	Keterangan
1800	3E 00	LD A, 0	A 00 H
1802	06 01	LD B, 1	B 01 H
1804	0E 02	LD C, 2	C 02 H
1806	16 03	LD D, 3	D 03 H
1808	1E 04	LD E, 4	E 04 H
180A	26 05	LD H, 5	H 05 H
180C	2E 06	LD L, 6	L 06 H
180E	FF	RST 38H	Keprogram monitor

2. Program bahasa *assembly* ditulis untuk menyusun isi *register* berikut :
   B = 12, C = 34, D = 56, E = 78, H = 9, L = A (gunakan intruksi LD 16 bit untuk transfer 1 byte data)

Alamat	*Bahasa Mesin*	Bahasa *Assembly*	Keterangan
1820	*01 34 12*	LD BC, 1234 H	B 12 H; C 34H
1823	*11 78 56*	LD DE, 5678 H	D 56 H; E 78H
1826	*21 0A 09*	LD HL, 09 0A H	H 09 H; L 0AH
1829	*FF*	RST 38H	Ke program monitor

4. Program dalam contoh 3 diterjemahkan ke dalam bahasa mesin dan ditampilkan dalam MPF. Kemudian program tersebut dijalankan dan dicek apakah isi dari alamat 1850H – 186FH telah clear (00). Jika tidak, program dicek dan dijalankan lagi.

Alamat	Bhs. Mesin	Label	Mnemonik	Keterangan
1800	06 20		LD B, 20 H	;*Set loop counter* = 32
1802	21 50 18		LD HL, 1850 H	;Set HL= alamat memori; mulai di hapus (00)
1805	AF		XOR A	;Set A = 0
1806	77	LOOP	LD (HL), A	;Masukkan 0 ke alamat; memori yang ditunjukkan
1807	23		INC HL	oleh HL
1808	05		DEC B	;tambah HL dengan 1
1809	20 FA		JRNZ LOOP	Kurangi B dengan 1 Jika B tidak = 0, kembali
180B	FF		RST 38H	ke LOOP Ke program monitor

5. Program bahasa *assembly* ditulis untuk menyusun isi dari alamat memori 1840H-184FH sebagai berikut : 0,1,2,3,…,F

Alamat	Bhs. Mesin	Label	Mnemonic	Keterangan
1800	06 10		LD B, 10 H	;*Set loop counter* = 32
1802	21 40 18		LD HL, 1840 H	;Set HL= alamat memori; mulai di hapus(00)
1805	AF		XOR A	;Set A = 0
1806	77	LOOP	LD (HL), A	;Masukkan 0 ke alamat; memori yang ditunjukkan
1807	23		INC HL	oleh HL
1808	3C		INC A	;tambah HL dengan 1
1809	05		DEC B	Tambah A dengan 1
180A	20FA		JRNZ LOOP	Kurangi B dengan 1 Jika B tidak = 0, kembali
180C	FF		RST 38H	ke LOOP Ke program monitor

Aplikasi dasar operasi aritmetik dan logik

1. Program menjumlahkan isi *register* D dan isi *register* E bersama-sama, hasilnya akan pada pasangan *register* HL.

Alamat	Op. Code	Mnemonik	Keterangan
		ORG. 1800H	; ADDRESS AWAL
1800	IE 8B	LD E, N	; N MASUK KE REG. A
1803	16 5A	LD D,N	; N MASUK KE REG. A
1804	7B	LD A.E	; E MASUK KE REG. A
1805	82	ADD A,D	; A+D MASUK KE A
1806	6F	LD L,A	; A MASUK KE REG. A
1807	3E 00	LD A,0	; 0 MASUK KE REG. A
1808	CE 00	ADC A,0	; A + 0 + CARRY
1809	67	LD H,A	; A MASUK KE REG. H
180A	FF	RST 38H	; KEMBALI KE MONITOR

Nilai Preset		Hasil Program				
Register		*Register*	*Flag*			
D	E	HL	Sign	Zero	P/V	Carry
5AH	A6H	01 00	0	0	0	0
46H	77H	00 BD	0	1	0	0

2. Program menjumlahkan data 16 bit di memori pada alamat 1A00H –1A01H pada data 16 yang ada pada pasangan register DE. Hasilnya akan disimpan pada pasangan *register* HL.

Alamat	*Op. Code*	*Mnemonic*	Keterangan
		ORG. 1800H	; ADDRESS AWAL
1800	3A 00 1A	LD A,(1A00H)	; 1A00H MASUK KE
1804	83	ADD A,E	REG. A
1805	6F	LD L,A	; A + E
1806	3A 01 1A	LD A,(1A01H)	; A MASUK KE REG. L
1809	8A	ADC A,D	; 1A01H MASUK KE A
180A	67	LD H,A	; D MASUK KE REG. A
180B	FF	REST 38H	; A MASUK KE REG. H
			; KEMBALI KE MEMORI.

Nilai *Preset*		Hasil Program				
Register		*Register*	*Flag*			
(1A01H)	(1A00H)	DE	Sign	Zero	P/V	Carry
8D	4F	46 47	1	0	0	0

Sumber dari Guru Mikro Saya. Inelco Bandung (1986) dan Laventhal, (1986). *Z80 Assembly Language Programming,* Mc Graw Hill, Singapore.

3. Program untuk menambahkan data 32 bit di memori pada address 1A00H – 1A03H kepada data 32 bit di memori pada address 1A04H – 1A07H. Hasilnya disimpan di memori pada address 1A08H – 1A0BH. Byte orde tinggi disimpan pada address yang lebih tinggi.

Alamat	Op. Code	Mnemonik	Keterangan
		ORG. 1800H	; ADDRESS AWAL
1800	06 04	LD B,4	; 4 MASUK KE REG. B
1802	DD 21 00 1A	LD IX, 1A00H	; 1A00H MASUK KE REG.
1806	A7	AND A	IX
1807	LOOP DD 7E	LD A,IX	
180A	00	ADC A,(IX+4)	; 1IX MASUK KE A
180D	DD 8E 04	INC IX	; (IX+4) + A + CARRY
180F	DD 23	JP NZ, LOOP	; REG.(IX + 1)
	C2 07 18		; JIKA TIDAK 0 KEMBALI
1810		REST 38H	KE
	FF		LOOP JIKA 0
			LANJUTKAN
			; KEMBALI KE MONITOR

Nilai Preset	Hasil Program					
Memori	Memori	Memori	Flag			
(1A03H-1A00H) 01 02 03 40	(1A00H-1A04H) 05 06 07 08	(1A0BH-1A08H) 06 08 0A 0C	Sign 1	Zero 0	P/V 0	Carry 0

4. Contoh program suatu sistem mikroprosesor digunakan sebagai Z80 *multivibrator*

```
        ORG     0000H
        LD      SP,0FFFFH
PPI1    EQU     37H
PORTB1  EQU     35H
        LD      A,90H
        OUT     (PPI1),A
LOOP:   LD      A,0
        OUT     (PORTB1),A
        CALL    TIMER1
        LD      A,0FFH
        OUT     (PORTB1),A
        CALL    TIMER1
        JP      LOOP
TIMER1:         LD      E,0AH
J50:    LD      B,0FFH
```

```
J51:    LD      D,0FFH
J52:    DEC     D
        JP      NZ,J52
        DEC     B
        JP      NZ,J51
        DEC     E
        JP      NZ,J50
        RET
        END
```

5. Program untuk menggeser isi data ke kiri pada lokasi memori (2000) dan hasilnya disimpan pada lokasi memori (2001).

Alamat Memori	Op - Code	Mnemonics	Keterangan
1800	3A	LD A, (2000)	data dikopi ke A
1801	00		dari alamat 2000
1802	20		
1803	B7	OR A	Membersihkan CF pada A.
1804	27	SL A	Isi A geser ke kiri.
1805	32	LD (2001), A	Simpan isi A ke lokasi memori (2001).
1806	01		
1807	20		
1808	C7	RST 00	Kembali ke alamat 00.

6. Program menampilkan nama Yoyo pada trainer mikroprosesor Z80.

Alamat Memori	Op - Code	Mnemonics	Keterangan
1800	DD	LD IX, HELP	Menyimpan pada
1801	21		alamat 1820
1802	20		
1803	18		
1804	CD	CALL, Scann	Memanggil scan
1805	FE		
1806	05		
1807	FE	CP 13, Key Step	Membandingkan
1808	13		dgn, Key step.
1809	20	JR NZ, Disp	Loncat jika tidak
180A	F9		sama dgn nol.
1820	BD	HELP.	
1821	B6		
1822	BD		
1823	B6		

7. Program nama menampilkan nama yoyo bergerak kekiri atau kekanan bergantian pada *trainer* mikroprosesor Z 80

Programnya sebagai berikut :

Alamat memori	Op-Code	Label/Keterangan	Mnemonics
1800	0606		LD B, 06
1802	21 00 19		LD HL, 1900
1805	11 00 1B		LD DE, 1B00
1808	1A	Loop	LD A, (DE), Loop
1809	D3 01		OUT (01), A
180B	7 E		LD A, (HL)
180C	D3 02		OUT (02), A
180E	CD 00 1C		CALL, 1C00.
1811	1C		INC E
1812	2C		INC L
1813	10 F3		DJNZ, Loop
1815	C3 00 18		JP 1800.
Subroutine			
1900	C1 E0	Pengatur arah	
1901	C2 D0	Huruf	
1902	C2 C8		
1903	C8 C4		
1904	D0 C2		
1905	E0 C1		
1B00	BD	Huruf yang	
1B01	B6	Ditampilkan	
1B02	BD	adalah Yoyo	
1B03	B6		
1B04	40		
1C00	3F 00		LD A, 00
1C 02	0F FF	Untuk mengatur	LD C, FF
1C04	0D	waktu	DEC C
1C05	20 FD		JR NZ
1C07	3D		DEC A
1C08	20 F8		JR NZ
1C0A	C9		RET

8. Program untuk membuat *time delay* 1 mili detik dengan Frekuensi *clock* mikroprosesor Z 80 misalkan 4 M Hz. , maka Periodenya (T) = 0, 25 µ detik, yaitu dari (T) = 1/ 4 M Hz = 10^{6}/ 4 = 0, 25 µ detik.

   Dengan siklus mesin = 1 m detik / 0,25 µ detik. = 1000/ 0,25 = 4000 siklus.

   Dari 7 + (4 + 12) x MCS + 4 = 4000 siklus

   11 + 16 MCS = 4000 = MCS = 3989/16 = F9.

   Maka program lengkapnya seperti dibawah ini

   Angka 7 adalah banyaknya siklus untuk LD r, n dari data book.

   Angka 4 adalah banyaknya siklus untuk DEC r.

Angka 12 adalah banyaknya siklus untuk JR, NZ., e.

Angka 4 adalah banyaknya siklus untuk RET.

MCS adalah data yang harus dicari.

Alamat memori	Op-Code	Label/Keterangan	Mnemonics
1800	OE	Delay	LD C, MCS
1801	MCS		
1802	OD	Dly1	DEC C
1803	20		JR NZ, FD
1804	FD		
1805	3D		DEC A
1806	20		JR NZ., Delay
1807	F8		
1808	C9		RET

Sumber: Laventhal, (1986). *Z80 Assembly Language Programming,* Mc Graw Hill, Singapore.

10. Perbandingan Instruksi Z 80 dengan Intel 8080/ 8085

Tabel 12. Perbandingan Instruksi Z80 dengan 8080/8085

Sumber : Harry Garland. (1979). *Introduction to microprocessor system design.* New Jersey : MC Graw Hill.

Z80	8080/ 8085	Keterangan
LD r1, r2	MOV r1, r2	Pindahkan dari *register* ke *register*
LD (HL), r	MOV M, r	Pindahkan dari *register* ke memori
LD r, (HL)	MOV r, M	Pindahkan dari memori ke *register*
HALT	HLT	Berhenti
LD r, n	MVI r, n	Pindahkan segera ke *register*
LD (HL), n	MVI M, n	Pindahkan segera ke memori
INC r	INR r	Penambahan *register*
DEC r	DCR r	Pengurangan *register*
INC (HL)	INR M	Penambahan memori
DCR (HL)	DCR M	Pengurangan memori
ADD A, r	ADD r	Tambahkan register ke A
ADC A, r	ADC r	Tambahan *register* ke A dengan pindahan (*carry*)
SBC A, r	SBB r	Kurangkan *register* dari A dengan pinjaman
AND r	ANA r	AND *) *register* dengan A
XOR r	XRA r	Exclusive OR*) Register dengan A
OR r	ORA r	OR*) *Register* dengan A
CP r	CMP r	Bandingkan *register* dengan A
ADD A, (HL)	ADD M	Tambahkan memori ke A

ADC A, (HL)	ADC M	Tambahkan memori ke A dengan pindahan
SUB A, (HL)	SUB M	Kurangkan memori dari A
SBC A, (HL)	SBB M	Kurangkan memori dari A dengan pinjaman
AND A, (HL)	ANA M	Dan memori dengan A
XOR A, (HL)	XRA M	Exclusive OR* memori dengan A
OR A, (HL)	OR*	Memori dengan A
OR A, (HL)	ORA M	OR* Memori dengan A
CP A, (HL)	CMP M	Bandingkan memori dengan A
ADD A, n	ADI n	Tambahkan segera ke A
ADC A, n	ACI n	Tambahkan segera ke A dengan pindahan
SUB A, n	SUI n	Kurangkan segera dari A
SBC A, n	SBI n	Kurangkan segera dari A dengan pinjaman
AND A, n	ANI n	AND* segera dengan A
XOR A, n	XRI n	*Exclusive* OR segera dengan A
OR A, n	ORI n	OR segera dengan A
CP A, n	CPI n	Bandingkan segera dengan A
RLCA	RLC	Putar A ke kiri
RRCA	RRC	Putar A ke kanan
RLA	RAL	Putar A ke kiri melalui pindahan
RRA	RAR	Putar A ke kanan melalui pindahan
Jp	JMP	Loncat tak bersyarat
JP C	JC	Loncat bila ada pindahan
JP NC	JNC	Loncat bila tak ada pindahan
JP Z	JZ	Loncat jika sama dengan nol
JP NZ	JNZ	Loncat jika tak sama dengan nol
JP P	JP	Loncat positif
JP M	JM	Loncat bila minus
JP PE	JPE	Loncat jika paritas genap
JP PO	JPO	Loncat jika paritas ganjil
CALL	CALL	Panggil tanpa syarat
CALL C	CC	Panggil bila ada pindahan
CALL NC	CNC	Panggil bila tidak ada pindahan
CALL Z	CZ	Panggil bila sama dengan nol
CALL NZ	CNZ	Panggil bila tidak sama dengan nol
CALL P	CP	Panggil bila positif
CALL M	CM	Panggil bila minus
CALL PE	CPE	Panggil bila paritas genap
CALL PO	CPO	Panggil bila paritas ganjil
RET	RET	Kembali
RET C	RC	Kembali bila ada pindahan
RET NC	RNC	Kembali bila tidak ada pindahan
RET Z	RZ	Kembali bila nol
RET NC	RNZ	Kembali bila tidak ada nol
RET P	RP	Kembali bila positif
RET M	RM	Kembali bila minus
RET PE	RPE	Kembali bila paritas genap
RET PO	RPO	Kembali bila paritas ganjil
RST	RST	Mulai kembali

IN A, (n)	IN n	Masukan dari pintu n
OUT (n), A	OUT n	Keluaran dari pintu n
LD BC, nn	LXI B, nn	Isi segera pasangan *register* B dan C
LD DE, nn	LXI D, nn	Isi segera pasangan *register* D dan E
LD HL, nn	LXI H, nn	Isi segera pasangan *register* H dan L
LD SP, nn	LXI SP, nn	Isi segera penunjuk tumpukan (SP)
PUSH BC	PUSH B	Dorong pasangan *register* B & C pada tumpukan
PUSH DE	PUSH D	Dorong pasangan *register* D & E pada tumpukan
PUSH HL	PUSH H	Dorong pasangan *register* H & L pada tumpukan
PUSH AF	PUSH PSW	Dorong A dan bendera-bendera pada tumpukan
POP BC	POP B	Sembulkan pasangan register B &C dari tumpukan
POP DE	POP D	Sembulkan pasangan *register* D &E dari tumpukan
POP HL	POP H	Sembulkan pasangan *register* H &L dari tumpukan
POP AF	POP PSW	Sembulkan pasangan A dan bendera-bendera dari tumpukan
LD (nn), A	STA nn	Simpan A langsung
LD A, (nn)	LDA nn	Isi A langsung
EX DE, HL	XCHG	Tukar *register-register* D & E dan H & L
EX (SP), HL	XTHL	Tukar puncak dari tumpukan H&L
LD SP, HL	SPHL	H & L ke penunjuk tumpukan (SP)
JP (HL)	PCHL	H & L ke penunjuk program
ADD HL, BC	DAD B	Tambahkan B & C ke H & L
ADD HL, DE	DAD D	Tambahkan D & E ke H & L
ADD HL, HL	DAD H	Tambahkan H & L ke H & L
ADD HL, SP	DAD SP	Tambahkan penunjuk tumpukan ke H & L
LD (BC), A	STAX B	Simpan A tak langsung
LD (DE), A	STAX D	Simpan A tak langsung
LD A, (BC)	LDAX B	Isi A tak langsung
LD A, (DE)	LDAX D	Isi A tak langsung
INC BC	INX B	Penambahan *register-register* B&C dengan satu
INC DE	INX D	Penambahan *register-register* D&E dengan satu
INC HL	INX H	Penambahan *register-register* H&L dengan satu
INC SP	INX SP	Penambahan penunjuk tumpukan (SP)
DEC BC	INX B	Penurunan B & C
DEC DE	INX D	Penurunan D&E
DEC HL	INX H	Penurunan H&L
DEC SP	INX SP	Penurunan penunjuk tumpukan (SP)
CPL	CMA	Komplemen A
SCF	STC	Pasang pindahan
CCF	CMC	Komplemen pindahan
DAA	DAA	Penyesuaian decimal A
LD (nn), HL	SHLD nn	Simpan H & L langsung
LD HL, (nn)	LHLD nn	Isi H & L langsung
EI	EI	Menjalankan interupsi
DI	DI	Melumpuhkan interupsi
NOP	NOP	Tidak ada operasi

11. Soal Latihan

1. Buat program dalam bahasa *assembly* Z 80 untuk menstransfer data dari lokasi memori 1900 ke *register* D dan 1901 ke *register* E!
1. Buat *flowchart* dan program dalam bahasa *assembly* untuk menjumlahkan bilangan yang berada pada lokasi memori 2000 dengan 2001 beserta carinya, bila (2000 = FD) dan (2001 = CB). Simpan hasilnya pada lokasi memori 2002 !
2. Buat *Flowchart* dan program dalam bahasa *assembly* untuk mengurangkan bilangan yang berada pada *register* H , dikurangi oleh bilangan yang berada pada lokasi memori 1900. Simpan hasilnya pada lokasi memori 1901. Data pada *register* H = 09, data pada (1900) = 0A !
3. Tuliskan *fiowchart* dan program bahasa *assembly* untuk menghapus isi memori pada adres 1950 sampai dengan 1965. Gunakan register B sebagai penghitung *loop* dan *register* HL sebagai penunjuk *address* memori !
4. Bilangan secara seri (tiga buah) yang berada pada lokasi memori 0041. Deret bilangan dimulai dari lokasi memori 0042. Simpan hasil dari jumlah bilangan tersebut dilokasi memori 0040. Diasumsikan bilangan 8 bit !
   Hasil (0040) = (0042 = 28) + (0043 = 53) + (0044 = 26).
5. Jumlahkan bilangan yang berada pada lokasi memori 1A00 – 1A03 dengan 1A04 – 1A07. Simpan hasilnya pada 1A08 – 1A0B?. Buat program dan *flowchart* dalam bahasa *Assembly* Z 80!
6. Buat *flowchart* dan program dalam bahasa *assembly* Z 80 untuk mengosongkan data yang berada pada Akumulator, dengan data A7. Dan simpan hasilnya pada lokasi memori 1900?

13. Referensi :

a. Harry Garland. (1979). *Introduction to microprocessor system design.* New Jersey : MC Graw Hill.
b. Laventhal, (1985). *Instroduction to microprocessor; software, hardware, programming.* Prentice Hall.
c. Laventhal, (1986). *Z80 Assembly Language Programming,* Mc Graw Hill, Singapore.
d. Inelco, (1986). *Guru Mikro Saya.*