IDENTIFIKASI JENIS KODE DALAM INSTRUCTION SET X86 FAMILY

Disusun Guna Memenuhi Tugas Organisasi dan Arsitektur
Komputer Semester V Pengampu: Bana Handaga, Dr. Ir, M.T

Oleh:

Muhammad Nunnizar Farichi
L200150072

PROGAM STUDI INFORMATIKA
FAKULTAS KOMUNIKASI DAN INFORMATIKA
UNIVERSITAS MUHAMMADIYAH
SURAKARTA 2017

SET INSTRUKSI

- **Pengertian Set Instruksi**

  Set Instruksi (Instruction Set atau Instruction Set Architecture (ISA)) didefinisikan sebagai suatu aspek dalam arsitektur komputer yang dapat dilihat oleh para pemrogram. Secara umum, ISA ini mencakup jenis data yang didukung, jenis instruksi yang dipakai, jenis register, mode pengalamatan, arsitektur memori, penanganan interupsi, eksepsi, dan operasi I/O eksternalnya (jika ada). ISA merupakan sebuah spesifikasi dari kumpulan semua kode-kode biner (opcode) yang diimplementasikan dalam bentuk aslinya (native form) dalam sebuah desain prosesor tertentu. Kumpulan opcode tersebut, umumnya disebut sebagai bahasa mesin (machine language) untuk ISA yang bersangkutan. ISA yang populer digunakan adalah set instruksi untuk chip Intel x86, IA-64, IBM PowerPC, Motorola 68000, Sun SPARC, DEC Alpha,dan lain-lain. ISA kadang-kadang digunakan untuk membedakan kumpulan karakteristik yang disebut di atas dengan mikroarsitektur prosesor, yang merupakan kumpulan teknik desain prosesor untuk mengimplementasikan set instruksi (mencakup microcode, pipeline, sistem cache, manajemen daya, dan lainnya).

  Pada beberapa mesin, semua instruksi memiliki panjang yang sama, pada mesin-mesin yang lain mungkin terdapat banyak panjang berbeda. Instruksi-instruksi mungkin lebih pendek dari, memiliki panjang yang sama seperti, atau lebih panjang dari panjang word. Membuat semua instruksi memiliki panjang yang sama lebih muda dilakukan dan membuat pengkodean lebih mudah tetapi sering memboroskan ruang, karena semua instruksi dengan demikian harus sama panjang seperti instruksi yang paling panjang.

- **Elemen-Elemen dalam Instruksi**

  1. Operation code (op code): Menentukan operasi yang akan dilaksanakan.
  2. Source operand reference: Merupakan input bagi operasi yang akan dilaksanakan.
  3. Result operand reference: Merupakan hasil dari operasi yang dilaksanakan.
  4. Next instruction preference: Memberitahu CPU untuk mengambil instruksi berikutnya setelah instruks yang dijalankan selesai.

- **Jenis-Jenis Instruksi**

  1. Data Processing (Pengolahan Data) : Instruksi-instruksi aritmetika dan logika. Instruksi aritmetika memiliki kemampuan untuk mengolah data numeric, sedangkan instruksi logika

beroperasi pada bit-bit word sebagai bit bukan sebagai bilangan. Operasi-operasi tersebut dilakukan terutama untuk data di register CPU.

2. Data Storage (Penyimpanan Data): Instruksi-instruksi memori. Instruksi-instruksi memori diperlukan untuk memindah data yang terdapat di memori dan register.
3. Data Movement (Perpindahan Data): Instruksi I/O. Instruksi-instruksi I/O diperlukan untuk memindahkan program dan data ke dalam memori dan mengembalikan hasil komputansi kepada pengguna.
4. Control (Kontrol): instruksi pemeriksaan dan percabangan. Instruksi-instruksi control digunakan untuk memeriksa nilai data, status komputansi dan mencabangkan ke set instruksi lain.

SET INSTRUKSI INTEL

1. Set Instruksi pada Mikroprosessor

No.	Tipe	Instruksi	
		Nama	**Fungsi**
1	Transfer data	MOVE	Mentransfer data dari lokasi sumber ke lokasi tujuan
		LOAD	Mentransfer data dari lokasi memori ke register CPU
		STORE	Mentransfer data dari register CPU ke lokasi memori
		PUSH	Mentransfer data dari sumber ke stack
		POP	Mentransfer data dari stack ke tujuan
		XCHG	Saling menukar isi sumber dan tujuan
		CLEAR	Me-reset tujuan dengan semua bit ‘0’
		SET	Mengeset tujuan dengan semua bit ‘1’
2	Aritmatika	ADD	Penjumlahan, hitung jumlah dari 2 operan
		SUB	Pengurangan, hitung selisih dari 2 operan
		MUL	Perkalian, hitung hasil kali dari 2 operan
		DIV	Pembagian, hitung hasil bagi dari 2 operan
		NEG	Negasi, ganti tanda operan

		INC	Tambahkan 1 pada operan
		DEC	Kurangkan 1 dari operan
		SHIFT A	Geser operan (kekiri atau kekanan) dengan tanda
3	Logika	NOT	Komplemenkan (komplemen 1) operan
		OR	Lakukan operasi logika OR pada operan
		AND	Lakukan operasi logika AND pada operan
		XOR	Lakukan operasi logika XOR pada operan
		SHIFT	Geser operan (kekiri atau kekanan), isi nilai pada ujung bit
		ROL/ROR	Geser operan (kekiri atau kekanan) dengan berputar
		TEST	Uji kondisi yang ditetapkan dan pengaruhi flag yang sesuai
4	Kendali Transfer	JUMP	Perpindahan tak bersyarat, masukkan alamat yang ditetapkan ke PC
		JUMPIF	Perpindahan bersyarat, masukkan alamat yang ditetapkan ke PC jika kondisi terpenuhi
		JUMPSUB	CALL, simpan 'status program control' yang sekarang, pindah kealamat yang ditetukan ke PC
		RET	RETURN, restore 'status program control' dari stack ke PC dan register/flag yang relevan lainnya
		EXECUTE	Mengambil operand dari lokasi tertentu dan mengeksekusi sebagai instruksi.
		SKIP	Menambah PC sehingga melompati instruksiberikutnya.
		SKIP BERSYARAT	Melompat atau tidak melakukan apa-apa berdasarkan pada persyaratan.
		HALT	Menghentikan eksekusi program.
		WAIT (HOLD)	Melanjutkan eksekusi pada saat persyaratan dipenuhi.
		NO OPERATION	Tidak ada operasi yang dilakukan.

		SKIP BERSYARAT	Melompat atau tidak melakukan apa-apa berdasarkan pada persyaratan.
5	Input/Output	IN (read)	Mentransfer data dari perangkat atau port i/o yang ditentukan ke tujuan (memori utama atau register)
		OUT (write)	Mentransfer data dari sumber yang ditentukan ke perangkat atau port i/o
		START I/O	Mentransfer instruksi ke prosesor i/o untuk menginisiasi operasi i/o
		TEST I/O	Mentransfer informasi status dari sistem i/o ke instruksi yang ditentukan
6	Konversi	TRANSLATE	Menterjemahkan nilai-nilai dalam suatu bagian memori berdasarkan tabel korespodensi
		CONVERT	Mengkonversi isi suatu word dari suatu bentuk ke bentuk lainnya (contoh decimal ke biner)
5	Input/Output	IN (read)	Mentransfer data dari perangkat atau port i/o yang ditentukan ke tujuan (memori utama atau register)
		OUT (write)	Mentransfer data dari sumber yang ditentukan ke perangkat atau port i/o
		START I/O	Mentransfer instruksi ke prosesor i/o untuk menginisiasi operasi i/o

INSTRUCTION SET PROCESSOR INTEL

- 4004
- 8008 / Datapoint 2200
- 8080 (111 Instructions), 8085 (113 Instructions)

1. DATA TRANSFER INSTRUKSI

Nama	Fungsi
MOV Rd, Rs	Mengcopy nilai dari Rs ke Rd
MOV Rd, M	Mengcopy nilai dari M ke Rd
MOV M, Rs	Mengcopy nilai dari M ke Rs
MVI Rd, d8	Memindahkan nilai register d8 ke register d8
MVI M, d8	Memindahkan nilai register d8 ke register M
LDA addr16	Menyalin data memori pada alamat yang spesifik addr16
LDAX rp	Mengcopy data pada register pair (rp)
LXI rp, d16	Mengisi register pair (rp) dari nilai data d16 (alamat 16 bit)
LHLD addr16	Menyalin data memori pada alamat yang spesifik addr16
STA addr16	Menyimpan nilai data langsung dalam memori addr16
STAX	Menyimpan nila data pada alamat register pair (rp)
SHLD addr16	Menyimpan data register H & L langsung dalam memori alamat addr16
SPHL	Memindahkan isi dari H & L ke pointer stack
XCHG	Menukar register H & L dengan register D & E
XHTL	Menukar stack tertinggi dengan register H & L
PUSH rp	Push 2 byte data ke stack pada register pair (rp)
PUSH PSW	Push 2 byte data ke stack pada processor status word (8-bit)
POP rp	Pop Two Bytes of Data off the Stack

2. ARITHMETIC INSTRUKSI

Nama	Fungsi
ADD reg	Instruksi penambahan pada register reg
ADD M	Instruksi penambahan pada register M
ADI d8	Instruksi penambahan data secara immediate pada register d8
ADC reg	Instruksi penambahan menggunakan carry flag pada register reg
ADC M	Instruksi penambahan menggunakan carry flag pada register M

ACI d8	Instruksi penambahan data d8 secara immediate menggunakan carry
DAA	Instruksi untuk mengatur bentuk desimal
DAD rp	Penambahan register pair ganda ke H & L register pair (rp)
SUB reg	Instruksi pengurangan pada register reg
SUB M	Instruksi pengurangan pada register M
SUI d8	Instruksi pengurangan data pada d8 secara immediate
SBB reg	Instruksi pengurangan menggunakan carry flag pada register reg
SBB M	Instruksi pengurangan menggunakan carry flag pada register M
SBI d8	Instruksi pengurangan secara immediate menggunakan carry flag pada register d8
INR reg	Instruksi kenaikan data reg setiap 1 byte
INR M	Instruksi kenaikan data M setiap 1 byte
INX rp	Instruksi kenaikan 1 data register pair (rp)
DCR reg	Instruksi penurunan data reg setiap 1 byte

3. LOGIKA INSTRUKSI

Nama	Fungsi
ANA reg	Menggunakan logika AND dengan logika accumulator pada data reg
ANA M	Menggunakan logika AND dengan logika accumulator pada data M
ANI d8	Menggunakan logika AND dengan logika accumulator immediate d8
ORA reg	Menggunakan logika OR dengan logika accumulator OR pada reg
ORA M	Menggunakan logika OR dengan logika accumulator OR pada M
ORI d8	Menggunakan logika OR dengan logika accumulator OR immediate register d8
XRA reg	Menggunakan logika eksklusif OR dengan logika accumulator eksklusif OR reg
XRA M	Menggunakan logika eksklusif OR dengan logika accumulator eksklusif OR reg M
XRI d8	Menggunakan logika eksklusif OR dengan data immediate pada register d8
CMP reg	Membandingkan data pada reg
CMP M	Membandingkan data pada register M
CPI d8	Membandingkan data secara immediate pada d8
CMA	Pelengkap accumulator data pada prosesor 8085
CMC	Pelengkap carry flag pada prosesor 8085

STC	Pengatur/set/setting carry flag
RLC	Pengatur rotasi/putaran accumulator pada bagian kiri

4. BRANCHING INSTRUKSI

Nama	Fungsi
JMP addr16	Membuat program beralih/loncat ke addr16
CALL addr16	Memanggil data pada addr16
RET	Kembali pada instruksi awal
RST n	Instruksi restart secara khusus
PCHL	Memindahkan H & L pada program counter

5. MACHINE CONTROL INSTRUKSI

Nama	Fungsi
SIM	Membuat settingan mask interrupt pada mesin prosesor 8085
RIM	Membaca mask interrupt pada mesin prosesor 8085
DI	Mengnonaktifkan system interrupt pada mesin prosesor 8085
EI	Mengaktifkan system interrupt pada mesin prosesor 8085
HLT	Memberhentikan mesin
NOP	Tidak ada operasi apapun pada kontrol mesin

- 8021 (66 Instructions)
- 8022 (73 Instructions)
- MCS-41 (8041) (87 Instructions)
- MCS-48 (8048) (93 Instructions)
- MCS-51 (8051)
- Intel iAPX 432
- Intel i860
- i960
- IA-64, Itanium, originated at Hewlett-Packard (HP), and later jointly developed by HP and Intel
- x86
  - IA-32 (i386, Pentium, Athlon)
  - Intel 64 64-bit version of x86, originally developed by AMD as AMD64
  - Extensions
    - FPU (x87) – Floating-point-unit (FPU) instructions

- MMX – MMX SIMD instructions
- MMX Extended – extended MMX SIMD instructions
- SSE – streaming SIMD extensions (SSE) instructions (70 instructions)
- SSE2 – streaming SIMD extensions 2 instructions (144 new instructions)
- SSE3 – streaming SIMD extensions 3 instructions (13 new instructions)
- SSSE3 – supplemental streaming SIMD extensions (16 instructions)
- SSE4.1 – streaming SIMD extensions 4, Penryn subset (47 instructions)
- SSE4.2 – streaming SIMD extensions 4, Nehalem subset (7 instructions)
- SSE4 – All streaming SIMD extensions 4 instructions (both SSE4.1 and SSE4.2)
- SSE4a – streaming SIMD extensions 4a (AMD)
- SSE5 – streaming SIMD extensions 5 (170 instructions)
- XSAVE – XSAVE instructions
- AVX – advanced vector extensions instructions
- FMA – fused multiply-add instructions
- AES – Advanced Encryption Standard instructions
- CLMUL – Carry-less mtiply (PCLMULQDQ) instruction
- Cyrix – Cyrix-specific instructions
- AMD – AMD-specific instructions (older than K6)
- SMM – System management mode instructions
- SVM – Secure virtual machine instructions
- PadLock – VIA PadLock instructions

Lebih jelasnya dapat dilihat pada tabel dibawah ini. Jenis operasi set instruksi x86

ACALL	Absolute Call	Memanggil subrutin program
ADD	Add Accumulator	InstruksiADDdigunakanuntukmelakukan penambahan pada dua buah operand. Dan destination (tempat hasil dari proses) selalu pada A, sdang operand source dapat berupa register, data langsung, maupun memory
ADDC	Add Accumulator (With Carry)	InstruksiADDdigunakanuntukmelakukan penambahan pada dua buah operand dengan carry
AJMP	Absolute Jump	AJMP ini adalah lompat tidak bersyarat jarak menengah. Disebut juga sebagai Jump 11-bit. Ini adalah

		instruksi 2-byte. Menjangkau alamat instruksi tepat di bawah AJMP, dan alamat label yang dituju, harus berada pada blok 2 KB yang sama.
ANL	AND Logic	Instruksi ini adalah melakukan AND logika pada dua operand dan menaruh hasilnya pada destination (Akumulator)
CJNE	Compare and Jump if Not Equal	Membandingkan data langsung dengan lokasi memori yang dialamati oleh register atau Akumulator jika tidak sama maka instruksi akan menuju ke alamat kode
CLR	Clear Register	Mereset isi register
CPL	Complement Register	Mengkomplement isi register
DA	Decimal Adjust	Mengkoreksi masalah yang timbul yang berkaitan denga penjumlahan bilangan BCD
DEC	Decrement Register	Mengurangi isi lokasi memori yang ditujukan oleh register R dengan 1, dan hasilnya disimpan pada lokasi tersebut
DIV	Divide Accumulator	Melakukan operasi pembagian
DJNZ	Decrement Register and Jump if Not Zero	Mengurangi nilai register dengan 1 dan jika hasilnya sudah 0 maka instruksi selanjutnya akan dieksekusi. Jika belum 0 akan menuju ke alamat kode
INC	Increment Register	Menambahkan isi memori dengan 1 dan menyimpannya pada alamat tersebut
JB	Jump if Bit Set	Membaca data per satu bit, jika data tersebut adalah 1 maka akan menuju ke alamat kode dan jika 0 tidak akan menuju ke alamat kode
JBC	Jump if Bit Set and Clear Bit	Membaca data per satu bit, jika data tersebut adalah 1, selain akan melompat ke instruksi lain juga akan menolkan bit yang baru saja diperiksa
JC	Jump if Carry Set	Membaca data carry
JMP	Jump to Address	Instruksi untuk memerintahkan menjangkau ke alamat kode tertentu

JNB	Jump if Bit Not Set	Membaca data per satu bit, jika data tersebut adalah 0 maka akan menuju ke alamat kode dan jika 1 tidak akan menuju ke alamat kode
JNC	Jump if Carry Not Set (jump if no carry, jump if CY=1)	Instruksi ini, menggunakan carry sebagai menentu keputusan dalam jump. Jika CY=1, maka program akan melompat ke alamat yang ditunjuk. Namun jika CY=0, maka program akan mengeksekusi instruksi selanjutnya dibawah JNC tersebut.
JNZ	Jump if Accumulator Not Zero	Instruksi ini tidak akan memeriksa isi A. Jika 00, maka program akan melompat ke alamat yang ditunjuk
JZ	Jump if Accumulator Zero	JZ (lompat jika A=0), atau JC (lompat jika CY=1), akan membuat program melompat pada lokasi yang ditunjuk hanya jika kondisi yang diminta terpenuhi
LCALL	Long Call	Ini adalah instruksi 3-byte. Byte pertama adalah opcode, sedang 2-byte lainnya adalah alamat 16-bit yang dituju. Saat instruksi LCALL ini dijalankan, CPU tidak lagi mengeksekusi instruksi-instruksi di bawah LCALL, namun segera melompat pada alamat yang dituju
LJMP	Long Jump	Instruksi 3-byte, di mana byte pertama adalah opcode, sedang dua byte yang lain adalah representasi dari alamat 16-bit yang dituju
MOV	Move Memory	Instruksi ini untuk memindahkan isi akumulator/register atau data dari nilai luar atau alamat lain
MOVC	Move Code Memory	Membedakan bahwa instruksi ini dipakai di memori program
MOVX	Move Extended Memory	Perintah yang dipakai untuk memori data eksternal
MUL	Multiply	Melakukan operasi perkalian
NOP	No Operation	Menyisipkan instruksi untuk tidak mengerjakan apa-apa
ORL	OR Logic	Operand tujuan dan sumber di-OR-kan, dan menempatkan hasilnya pada tujuan destination.

		Instruksi ORL dapat digunakan untuk men-Set menjadi 1′s beberapa bit dalam register.
POP	Pop Value From Stack	Memanggil subrutin dengan instruksi CALL, memory stack akan menyimpan alamat di mana CPU akan kembali setelah menjalankan subrutin
PUSH	Push Value Onto Stack	Memanggil subrutin dengan instruksi CALL, memory stack akan menyimpan alamat di mana CPU akan kembali setelah menjalankan subrutin
RET	Return From Subroutine	Intruksi untuk kembali dari suatu subrutin program ke alamat terakhir subrutin tersebut di panggil
RETI	Return From Interrupt	Mereset bit yang bersangkutan
RL	Rotate Accumulator Left	Pada putar kiri , 8-bit dalam akumulator digeser ke kiri sejauh satu bit. Bit D7 keluar dari Most Significant Bit (MSB) dan ditempatkan pada D0 atau Least Significant Bit (LSB)
RLC	Rotate Accumulator Left Through Carry	Menggeser 8-bit dalam akumulator ke kanan sejauh 1 bit melewati Carry. Bit D0 keluar dari Least Significant Bit (LSB) dan ditempatkan pada Carry. Sedangkan isi Carry ditempatkan pada D7 atau Most Significant Bit (MSB)
RR	Rotate Accumulator Right	Pada putar kanan , 8-bit dalam akumulator digeser ke kanan sejauh satu bit. Bit D0 keluar dari Least Significant Bit (LSB) dan ditempatkan pada D7 atau Most Significant Bit (MSB)
RRC	Rotate Accumulator Right Through Carry	Menggeser 8-bit dalam akumulator ke kanan sejauh 1 bit melewati Carry. Bit D0 keluar dari Least Significant Bit (LSB) dan ditempatkan pada Carry. Sementara isi Carry ditempatkan pada D7 atau Most Significant Bit (MSB)
SETB	Set Bit	Instruksi untuk mengaktikan atau memberikan logika 1 pada sebuat bit data

SJMP	Short Jump	SJMP adalah lompat tanpa syarat jarak pendek. Disebut juga sebagai Jump relatif 8-bit. Ini adalah instruksi 2 byte. Byte pertama adalah opcode, sedang byte lainnya adalah alamat relatif yang dituju. Ya alamat relatif, sehingga nilai pada byte ke dua ini bukan representasi dari alamat yang dituju, melainkan nilai relatif terhadap nilai PC saat itu
SUBB	Subtract From Accumulator With Borrow	SUBB bisa difungsikan sebagai SUB. Yaitu dengan membuat/memastikan CY=0 sebelum perintah SUBB dilaksanakan
SWAP	Swap Accumulator Nibbles	men-swap (menukar balik) nible bawah dan nible atas
XCH	Exchange Bytes	Merubah bit
XCHD	Exchange Digits	Merubah digit
XRL	Exclusive OR Logic	Membalik nilai (complement) beberapa bit tertentu dari sebuah bilangan biner 8 bit, caranya dengan membentuk sebuah bilangan biner 8 bit sebagai data konstan yang di-XRL-kan bilangan asal. Bit yang ingin dibalik-nilai diwakili dengan ‘1’ pada data konstan, sedangkan bit lainnya diberi nilai ‘0’

2. Set Instruksi pada Mikroprosessor 8086 dan 8088

Berikut ini penjelasan tentang instruksi yang digunakan pada bahasa rakitan 8086 dan 8088 atau sejenisnya :

No	Nama	Fungsi
1.	**AAA** (ASCII adjust for edition)	Pengaturan ASCII bagi penambahan
2.	**AAD** (ASCII adjust for division)	Pengaturan ASCII bagi pembagian
3.	**AAM** (ASCII adjust for multipy)	Pengaturan ASCII bagi perkalian
4.	**AAS** (ASCII adjust for substraction)	Pengaturan ASCII bagi pengurangan
5.	**ADC** (add with carry)	Tambahkan dengan carry
6.	**ADD** (addition)	Penambahan

7.	**AND** (logic AND)	Logik AND
8	**CALL** (CALL subroutine)	Subrutin panggil
9.	**CBW** (Convert byte to word)	Konversikan byte ke kata
10.	**CLC** (Clear Carry)	Kosongkan corry flag
11.	**CLD** (Clear directon flag)	Kosongkan flag arah
12.	**CLI** (Clear interrupt enable)	Kosongkan flag penggerak interupsi
13.	**DAS** (decimal adjust for substraction)	Pengaturan decimal bagi pengurangan
14.	**DEC** (decrement)	Penurunan operand tujuan dengan 1
15.	**DIV** (divide)	Pembagian tak bertanda
16.	**ESC** (escape)	Sehubungan dengan suatu co-procesor external
17.	**HLT** (halt)	Menghentikan prosesor sampai saluran reset diaktifkan
18.	**IDIV** (Integer division)	Pembagian kilat
19.	**IMUL** (Integer multiply)	Perkalian kilat
20.	**IN** (Input) IN (masukan)	Mentransfer data dari port yang disperifikasikan ke dalam register AC atau AX
21.	**NC** (Increment)	Menaikkan operand tujuan dalam 1
22.	**INT** (Interupt) INT (Interupsi)	Mengawali suatu prosedur interupsi dengan jenis yang dispesifikasikan oleh instruksi yang bersangkutan
23.	**INTO**	Instrupsi bila ada overflow INTO digunakan untuk membangkitkan suatu interupsi perangkat lunak yang bergantumg pada status flag OF
24.	**IRET** (Interupt return)	Kembali dari suatu interupsi dan mendapatkan kembali IP,CS dan flag-flag dari stack
25.	**JA** (Jump on above)	Jika operand pertama lebih besar daripada operand kedua

26.	**JNBE** (Jump)	Digunakan jika pada saat pembandingan operand 1, tidak lebih kecil atausama dengan operand 2.
27.	**JAE** (Jump on above or equal)	Digunakan pada saat pembandingan operand 1 lebih besar atau sama dengan operand 2.
28.	**JNB** (Jump not below)	Digunakan pada saat pembandingan operand 1 tidak lebih kecil dari operand 2.
29.	**JB** (Jump on below)	Digunakan pada saat operand 1 lebih kecil dari operand 2.
30.	**JNAE** (Jump on not above or equal)	Pada saat operand 1 tidak lebih besar atau sama dengan operand 2.
31.	**JBE** (Jump on below or equal)	Digunakan pada saat operand 1 lebih kecil atau sama dengan operand 2.
32.	**JNA** (Jump on not above)	Digunakan pada saat operand 1 tidak lebih besar dari operand 2
33.	**JC** (Jump on carry)	Digunakan pada saat akan diprogram CF=1.
34.	**JCXZ** (Jump if CX=0)	Digunakan bila isi register CX=0.
35.	**JE** (Jump on equal)	Digunakan pada saat pembandingan kedua operand sama.
36.	**JZ** (Jump on zero)	Digunakan pada saat akan diproses ZF=1
37.	**JG** (Jump on greather than)	Digunakan pada saat pembandingan operand 1 menunjukkan lebih besar dari operand 2
38	**JNLE** (Jump on greather or equal)	Digunakan pada saat operand 1tidak lebih kecil atau sama dengan operand 2
39.	**JGE** (Jump on greather or equal)	Digunakan pada saat operand 1 lebih besar atau sama dengan operand 2.

40.	**JNL** (Jump on less)	Digunakan pada saat operand 1 tidak lebih kecil dari operand 2.
41.	**JL** (Jump on less)	Digunakan pada saat operand 1 lebih kecil dari operand 2.
42.	**JNGE** (Jump on not greather or equal)	Digunakan pada saat operand 1 tidak lebih besar atau sama dengan operand 2.
43.	**JLE** (Jump on less or equal)	Digunakan pada saat operand 1 lebih kecil atau sama dengan operand 2.
44.	**JNG** (Jump on not greather than)	Digunakan pada saat operand 1 tidak lebih besar daripada operand 2
45.	**JMP** (Unconditional jump)	Lompatan tidak bersyarat
46.	**JNC** (Jump on not carry)	Digunakan pada saat diproses CF=O
47.	**JNE** (Jump on not equal)	Digunakan pada saat operand 1 tidak sama dari operand 2.
48.	**JNO** (Jump on not zero)	Digunakan pada saat diproses ZF=0
49.	**JNO** (Jump on not overflow)	Digunakan bila tidak ada overflow (OF=O).
50	**JNS** (Jump on not sign)	Digunakan pada saat diproses SF=O
51.	**ZNP** (Jump on not pority)	Digunakan bila tidak ada pointer
52.	**JPO** (Jump on poity odd)	Digunakan bila pointer ganjil
53.	**O** (Jump on overflow)	Digunakan pada saat ada overflow
54.	**JP** (Jump on Pority equal)	Digunakan bila pointer genap
55.	**JS** (Jump on sign)	Digunakan pada saat diproses ZF=1
56.	**LAHF** (Load AH from flag)	LAHF me-load bit 7,6,4,2 & 0 pada register AH masing-masing dengan isi flag SF.ZF,AF & CF.
57.	**LDS** (Load pointer using DS)	Muatkan peminjak dengan menggunakan DS.
58.	**LEA** (Load effective address)	LEA mentransfer opsan sumber 16 bit dalam memori ke tujuan 16 bit.
59.	**LES** (Load pointer using ES)	Muatkan pointer dengan menggunakan ES
60.	**LOCK** (Lock bas	Digunakan dalam penerapan pemakaian sumber bersama, untuk memastikan bahwa

		memori tidak diakses secara serentak oleh lebih dari satu proses.
61.	**LODS** (Load string)	Muatkan string byte atau kata
62.	**LOOP**	Loop jika EX bukan
63.	**LOOPE** dan LOOPZ -	-Loop bila sama
64.	**Loop** bila 0	bila 0
65.	**LOOPNE/LOOPNZ**	Loop bila tidak sama (ZF=0)
66.	**Loop**	bila tidak nol (ZF=O)
67.	**MOVE**	Pindahkan byte atau kata memindahkan data 8 bit atau 16 bit.
68.	**MOUS** (Move string)	Pindahkan data string 8 bit atau 16 bit
69.	**MUL** (Multifly)	Digunakan untukmengalihkan isi bertandapada akumulator dengan suatu operand sumber yang dispesifikasikan
70.	**NEG** (Negate)	Mengurangkan suatu operand tujuan dari 0 & menyimpan hasil komplemen keduanya dalam tujuan
71.	**NOP** (Logic NOT)	NOP tidak membe rikan dampak tertentu (tidak ada operasi)
72.	**NOT (**Logic OR)	Operasi logic OR
73.	**OR (**Logic OR)	Operasi logic OR
74.	**OUT** (Output)	Digunakan untuk mentransfer data dari AL atau AX ke point (pangkalan)
75.	**POP** (POP from stack)	Keluaran data dari stack dari AL atau AX ke point (pangkalan)
76.	**POPF** (POP flag)	Keluarkan ke flag dan stack
77.	**POSH** (Push to stack)	Dorong sumber ke stack
78.	**PUSHF** (Push flag)	Dorong flag ke stack
79.	**RCL** (Rotate lift with carry)	Putar ke kirin dengan carry 1
80	**RCR** (Rotate right with carry)	Putar ke kanan dengan carry 1
81.	**REP**	Ulangi
82.	**REE**	Ulangi jika sama
83.	**REPZ**	Ulangi jika nol

84.	**REPNE**	Ulangi jika tidak sama
85.	**REPNZ**	Ulangi jika tidak nol
86.	**RET** (Return from subroutine)	Digunakan untuk kembali ke subrutin
87.	**ROL** (Rotate left)	Rotasi (putar) ke kiri 1
89.	**SAHF** (store AH in flags register)	Simpan AH dalam reegister flag
90.	**SAL/SHL** (Shift left arithmetic/logical)	Geser aritmatik/logika ke kiri 1
91.	**SAR/SHR** (Shift right arithmetic/logical)	Geser aritmatik/logika ke kanan 1
92.	**SBB** (Substract with borrow)	Kurangkan dengan borrow
93.	**SCAS** (Scan string)	Digunakan untuk mengurangkan string tujuan yang dialamatkan a/n regiser DI,dari AX atau AL
94.	**STC** (Set carry flag)	Menset carry flag (CF)
95.	**STD** (Set direction flag)	Menset flag arah (DF)
96.	**STI** (Set interrupt enable flag)	Menset flag penggerak interupsi (IF)
97.	**STOS** (Substract)	Memindahkan operand sumber yang terkandung dalam AX atau AL ke suatu tujuan yang dialamatkan oleh DI
98.	**SUB** (Substract)	Mengurangkan sumber dari operand tujuan dan hasilnya ditepatkan dalam tujuan
99.	**TEST** (Logical comporison)	Menguji operand operand atau logic perbandingan
100	**WAIT**	Menunggu sampai saluran test aktif
101	**XCHG** (Exchange)	Pertukaran dan sumber
102	**XLAT** (Translate)	Digunakan untuk menerjemahkan se-byte dalam register AL ke dalam suatu byte yang diambil dari tabel terjemahan
103	**XOR** (Logic exclusive OR)	Operasi exclusive OR