

INSTRUCTION SET OF 8086

Classified into 7 categories

1] Data Transfer

2] Arithmetic

3] Logical

4] Control

5] Processor Control Instructions

6] String Manipulation

7] Interrupt Control

Data Transfer Instructions

Note : Data Transfer Instructions do not affect any flags

1] **MOV** dest, src

Note that source and destination cannot be memory location. Also source and destination must be same type.

2] **PUSH** Src : *Copies word on stack.*

3] **POP** dest : *Copies word from stack into dest. Reg.*

4] **IN** acc, port : *Copies 8 or 16 bit data from port to accumulator.*

a) Fixed Port

b) Variable Port

5] **OUT** port, acc

Data Transfer Instructions Contd ...

- 6] **LES** Reg, Mem : *Load register and extra segment register with words from memory.*
- 7] **LDS** Reg,Mem : *Load register and data segment register with words from memory.*
- 8] **LEA** Reg,Src : *load Effective address.*
(Offset is loaded in specified register)
- 9] **LAHF** : *Copy lower byte of flag register into AH register.*
- 10] **SAHF** : *Copy AH register to lower byte of flag register.*

Data Transfer Instructions Contd ...

11] **XCHG** dest, src : *Exchange contents of source and destination.*

12] **XLAT** : *Translate a byte in AL.*

This instruction replaces the byte in AL with byte pointed by BX. To point desired byte in look up table instruction adds contents of BX with AL ($BX + AL$). Goes to this location and loads into AL.

Arithmetic Instructions

- 1] **ADD** dest,src
- 2] **ADC** dest,src : *Add with carry*
- 3] **AAA** : *ASCII adjust after addition.*

We can add two ASCII numbers directly and use AAA after addition so as to get result directly in BCD. (**Works with AL only**)

- 4] **DAA** : *Decimal adjust accumulator.*
(**Works with AL only**)

Arithmetic Instructions Contd ...

5] **SUB** dest, src

6] **SBB** dest, src : *Subtract with borrow.*

7] **AAS** : *ASCII adjust for subtraction*

(Same as AAA and works with **AL** only)

8] **DAS** : *Decimal adjust after Subtraction.*

(**Works with AL only**)

9] **MUL** src

10] **IMUL** src : *Multiplication of signed byte.*

Arithmetic Instructions Contd ...

11] **AAM** : *BCD adjust after multiply.*

(**Works with AL only**)

12] **DIV** src

If any one attempts to divide by 0 , then ?

13] **IDIV** : *Division of signed numbers*

14] **AAD** : *BCD to Binary convert before
Division.*

15] **DEC** dest

Arithmetic Instructions Contd ...

16] **INC** dest

17] **CWD** : *Convert signed word to signed double word.*

18] **CBW** : *Convert signed byte to signed word.*

(**CBW** and **CWD** works only with **AL,AX** and **DX** registers.)

19] **NEG** dest : *Forms Twos complement.*

Logical Instructions

- 1] **AND** dest,src
- 2] **NOT** dest : *Invert each bit in destination*
- 3] **OR** dest,src
- 4] **XOR** dest,src
- 5] **RCL** dest,count : *Rotate left through Carry*
If rotate once count is directly specified in the instruction. For more no. of rotations count is specified in CL register.
- 6] **RCR** dest,count : *Rotate right through carry*
- 7] **ROL** dest,count : *Rotate left (into carry as well as into LSB)*
- 8] **ROR** dest,Count : *Rotate left (into carry as well as into MSB)*

Logical Instructions Contd...

- 9] **SAL/ SHL** dest, count : *Shift left and put 0 in LSB.*
- 10] **SAR** dest,count : *Shift right*
New MSB = Old MSB
- 11] **SHR** dest,count : *Shift right .MSB is filled with 0's.*
- 12] **TEST** dest,src : *AND logically,updates flags but source and dest are unchanged.*

Logical Instructions Contd...

13] **CMP** dest,src

CF, ZF and SF are used

Ex .CMP CX,BX

	CF	ZF	SF
CX = BX	0	1	0
CX > BX	0	0	0
CX < BX	1	0	1

CONTROL TRANSFER INSTRUCTIONS

1] **CALL** : *Call a procedure*

Two types of calls

i) Near Call (Intra segment)

ii) Far Call (Intersegment)

2] **RET** : *Return execution from procedure*

3] **JMP** : *Unconditional Jump to specified destination. Two types near and Far*

CONTROL TRANSFER INSTRUCTIONS Contd ...

4] **JA / JNBE** : *Jump if above / Jump if not below*

The terms above and below are used when we refer to the magnitude of Unsigned number .

Used normally after CMP.

5] **JAE / JNB / JNC**

6] **JB / JC / JNAE**

7] **JBE / JNA**

8] **JE/ JZ**

CONTROL TRANSFER INSTRUCTIONS Contd ...

9] **JCXZ** : *Jump if CX is Zero.*

10] **JG / JNLE** : *Jump if Greater /Jump if not less than or equal.*

The term greater than or less than is used in connection with two signed numbers.

11] **JGE / JNL** :

12] **JL / JNGE** :

13] **JLE / JNG** :

14] **JNE / JNZ** :

CONTROL TRANSFER INSTRUCTIONS Contd ...

15] **JNO** : *Jump if no overflow*

16] **JNS** : *Jump if no sign*

17] **JS**

18] **JO**

19] **JNP / JPO**

20] **JP / JPE**

In all above conditional instructions the destination of jump is in the range of -128 to + 127 bytes from the address after jump.

CONTROL TRANSFER INSTRUCTIONS Contd ...

21] **LOOP** : *Loop to the specified label if CX not equal to Zero.*

The count is loaded in CX reg. Every time LOOP is executed then CX is automatically decremented . (Used in delay programs.)

22] **LOOPE/ LOOPZ** : *Loop while CX not equal to zero and ZF = 1.*

23] **LOOPNE / LOOPNZ** : *Loop while CX not equal to zero and ZF = 0.*

In all above LOOP instructions the destination of jump is in the range of -128 to + 127 bytes from the address after LOOP.

PROCESSOR CONTROL

- 1] **CLC** : *Clear Carry flag.*
- 2] **STC** : *Set carry Flag*
- 3] **CMC** : *Complement Carry Flag*
- 4] **CLD** : *Clear Direction Flag.*
- 5] **STD** : *Set Direction Flag*
- 6] **CLI** : *Clear Interrupt Flag.*
- 7] **STI** : *Set Interrupt Flag.*
- 8] **HLT** : *Halt Processing.*

PROCESSOR CONTROL Contd...

9] **NOP** : *No Operation*

10] **ESC** : *Escape*

Executed by Co-processors and actions are performed according to 6 bit coding in the instruction.

11] **LOCK** : *Assert bus lock Signal*

This is **prefix** instruction.

12] **WAIT** : *Wait for test or Interrupt Signal.*

Assert wait states.

STRING CONTROL

1] **MOVS/ MOVSB/ MOVSW**

dest stringname ,src string name

This inst moves data byte or word from location in DS to location in ES.

2] **REP / REPE / REPZ / REPNE / REPNZ**

Repeat string instructions until specified conditions exist.

This is **prefix** instruction.

STRING CONTROL Contd...

3] **CMPS / CMPSB / CMPSW**

Compare string bytes or string words.

4] **SCAS / SCASB / SCASW**

Scan a string byte or string word.

Compares byte in AL or word in AX.

String address is to be loaded in DI.

5] **STOS / STOSB / STOSW**

Store byte or word in a string.

Copies a byte or word in AL or AX to memory location pointed by DI.

6] **LODS / LODSB / LODSW**

Load a byte or word in AL or AX

Copies byte or word from memory location pointed by SI into AL or AX register.

Interrupt Control

- 1] **INT** type
- 2] **INTO** *Interrupt on overflow*
- 3] **IRET** *Interrupt return*