Module II

Programming of 8086 Microprocessor

Instruction Set of 8086

Instruction set is divided in to 8 types :-

I. Data copy/transfer instructions: Used to transfer instruction from source operand to destination operand.move,load,exchange belong to this category

II. Arithmetic and Logic Instructions: Instructions performing arithmetic,logical,increment,decrement,compare belong to this category

III. Branch Instructions: Instructions that transfer control of execution to the specific address.Call,jump,interrupt and return instructions.

IV. Loop instructions: Instructions used to implement conditional and unconditional loops.LOOP,LOOPZ,LOOPNZ instructions belongs to this category

V. Machine control instructions: Instructions that control machine status.NOP,HLT,LOCK isntructions

VI. Flag manipulation instructions: Instructions that directly affect flag register.CLD,STD,CLI,STI belongs to this category

VII. Shift and rotate instructions: Instruction involving bitwise shifting or rotating in either direction

VIII. String Instuctions: Instruction involves string manipulation operations

Data Transfer Instructions

- MOV Instruction
- XCHG Intsruction
- XLAT Instruction
- LEA,LES,LDS Instruction

MOV Instruction: **MOV D,S**

The instruction have two operands, One source operand (s) and the destination operand (d). *The data will be transferred from the source operand to the destination operand leaving the data in the source operand unchanged.*

- MOV CX, 037AH Put immediate number 037AH to CX

XCHG instruction: **XCHG Destination, Source**

- The XCHG instruction exchanges the content of a register with the content of another register or with the content of memory location.

- **XCHG AX, DX** Exchange word in AX with word in DX

LEA Instructions: **LEA – Load Effective Address**

- **LEA Register, Source**
- This instruction determines the offset of the variable or memory location named as the **Source** and puts this offset in the indicated 16-bit **Register**.

LDS/LES Instruction: **LDS-Load pointer using DATA segment**
LES-Load pointer using EXTRA Segment

These two instructions are used to load the Data Segment(DS) or the Extra Segment (ES) registers and the register specified in the instruction from memory

LDS dest, src

Eg: LDS SI, ARRAY - Loads DS and SI from locations starting from offset ARRAY

XLAT Instruction: **Translate instruction**

- XLAT is used to translate a value from one coding system to another with the help of a look up table
- XLAT
  (AL) --- (BX)+(AL)
  Eg:-
- LEA BX,2000H
- MOV AL,20
- XLAT
- In this program AL will be loaded with the content of the memory location pointed by [2000+20]

Arithmetic Instructions: 8086 handles many arithmetic operation such as:-

- ➔ Addition
- ➔ Subtraction
- ➔ Multiplication
- ➔ Division
- ➔ Comparing two values
- ➔ Negation

In arithmetic instruction flag register is modified

ADD Instruction: ADD instruction add a data from source operand to a data from destination and save the result in destination operand.

ADD Destination,Source

Perform addition X+Y+Z

MOV AX,X
ADD AX,Y
ADD AX,Z

ADC instruction-Add with carry: This instruction adds the source operand to the destination operand along with carry flag

Result is stored in destination operand

```
MOV AX,5678H
ADD AX,4321H
MOV BX,1234H
ADC BX,FEDBH
```

SUB Instruction: This instruction subtracts the source operand from the destination operand and the result is stored in destination operand

➔ Flag registers are modified as per the result

➔SUB Destination ,Source

Eg : Subtraction of 03H from 05H

```
MOV AL,05H
SUB AL,03H
```

After execution AL will contain 02H

SBB-Subtract with borrow: This instruction subtracts the source operand from the destination operand along with carry flag and the result is stored in destination operand

ie after execution using SBB instruction :-

DEST operand = DEST operand - SOURCE operand – Carry flag

```
MOV AX,5678H
SUB AX,4321H
MOV BX,1234H
SBB BX,FEDBH
```

INC Instruction-*Increment:* ***INC instruction increment the operand by 1***

DEC Instruction – Decrement DEC instruction decrement the operand by 1

MUL instruction-**Unsigned Multiplication:** For 16 bit multiplication the product which is of 32 bits will be stored in DX:AX pair

- For 8 bit multiplication the product which is of 16 bit will be stored in AX **register**
- **MOV AL,FDH**
- **MOV CL,05H**
- **MUL CL**

IMUL Instruction-*Signed multiplication*

IMUL is same as that of MUL instruction except the fact that signed numbers are used here

NEG Instruction-Negate

NEG instruction is used to change the sign of register content or memory content

CMP Instruction:

CMP Reg1,Reg2

CMP Reg1,Memory

The content of reg or memory are compared by subtraction and the result is used to modify flag

Logical Instructions: Logical instructions are used for performing:-

1) AND
2) OR
3) Exclusive-OR
4) Complement
5) Shift and Rotate

AND Instruction: AND DEST,SOURCE: The content of source reg/memory/data is logically ANDed bit by bit and the result is stored in destination register/memory

AND BL,0FH

content of BL----- 0000 1010

OR Instruction:OR DEST,SOURCE : The contents are Ored bit by bit logically and the result is stored in destination register/memory

OR BL,0FH

XOR Instruction: XOR DEST, SOURCE :The content in the instruction is logically Exclusive Ored bit by bit and the result is stored in destination register/memory

Let content of BL is 1010

XOR BL,05H

The result is BL 1111

TEST Instruction: The content in the instruction are logically ANDed together and the result is used to modify flags**.** **TEST AL,01H**

We need to check only L.S.B, If this is 1, the number is odd, else the number is even

NOT Instruction: NOT operand

NOT Reg eg: NOT AL

The content of register/memory is complemented

SHIFT Instruction

They are of 3 types :

i) SHL /SAL– Shift Left/Shift Arithmetic Left Instruction
ii) SHR – Shift Right Instruction
iii) SAR – Shift Arithmetic Right Instruction

SHL/SAL

- ➔ The content of register/memory is shifted left
- ➔ The MSD is shifted to carry flag while the LSD is filled with zero

Eg:-SHL AX,1 – The content of AL is shifted towards left once

Eg:- SHL AX,CL – The content of AL is shifted towards left specified by the count value in CL register

SHR Instruction

- ➔ The content of register/memory is shifted right
- ➔ The LSD is shifted to carry flag
- ➔ The MSD is filled with Zero

Eg:- SHR AX,1 – *The content of AX is shifted right once*

Eg:- SHR AX,CL – The content of AX is shifted right specified by the count value in CL

SAR Instruction

- ➔ The content of register/memory is shifted right
- ➔ The LSD is shifted to Carry flag
- ➔ The MSD is retained

Eg:- SAR AX,1 – *The content of AX is shifted right once*

Eg:- SAR AX,CL- The content of AX is shifted right specified by count value in CL

Rotate Instructions

They are of four types :-

- ➔ R.O.L
- ➔ R.C.L
- ➔ R.O.R
- ➔ R.C.R

R.O.L Instruction: The content of register/memory is rotated left

- ➔ The M.S.D is moved to both LSD and Carry flag

Eg:-ROL BX,1 – *The content of register is rotated left once*

EG ROL BX,CL – *The content of BX is rotated left number of times as specified by count value in CL*

R.O.R Instruction: The content of register/memory is rotated right

- ➔ The L.S.D is moved to M.S.D and carry flag

Eg:- ROR BX,1
ROR BX,CL

R.C.R Instruction: The content of register/memory is rotated right Eg: RCR BX,1
RCR BX,CL

- ➔ The L.S.D is moved to carry flag
- ➔ The carry flag is moved to M.S.D

String Manipulation Instruction

String is a sequence of bytes or words.The 8086 includes following string instructions :-

- ➔ String Movement
- ➔ String comparison
- ➔ Load
- ➔ Scan
- ➔ Store

The string instruction ends with S (String) or SB(String byte) or SW (String word)

Instruction	Explanation
LODS	String data in DS is copied in to Accumulator register
LODSB	1 byte of string data in D.S is copied in to Accumulator register
LODSW	1 word of string data in D.S is copied in to Accumulator register
STOS	The content of accumulator register is stored as string data in to E.S
STOSB	The content of accumulator register is stored as 1 byte of string data in to E.S
STOSW	The content of accumulator register is stored as 1 word of string data in to E.S

Instruction	Explanation
REP or REPZ or REPE	When this instruction is used the string instruction execution is repeated until the content of CX = 0 or ZF = 0
REPNZ / REPNE	String instruction execution is repeated until CX = 0 or ZF = 1
MOVS	Used to copy string data in D.S to E.S
MOVSB	Used to copy one byte of string data in D.S to E.S
MOVSW	Used to copy one word of string data in D.S to E.S
CMPS	Used to compare / subtract string data in E.S from string data in D.S and the result is used to modify flags
CMPSB	Used to compare 1 byte of string data
CMPSW	Used to compare 1 word of string data
SCAS	String data in E.S is subtracted from the content of Accumulator and the result is used to modify flags
SCASB	1 byte of string data in E.S is subtracted
SCASW	1 word of string data in E.S is subtracted

Control Tranfer / Branch Instruction: Normally a program is executed sequentially.

- When control transfer instruction is encountered the execution control is transferred to specified destination.
- Th transfer of program execution is done by changing IP or IP and CS together
- The control transfer instruction consists of call, jump, loop and software interrupt instruction

Subroutines/Procedures: Subroutine / procedure is a program written on some memory location

- Each time when we use a branch instruction subroutine/procedure is executed
- ie Branch instruction will transfer control to location of subroutine

- When a task is to be done repeatedly then it is written as subroutine.
- Subroutine/procedure will be called each time to perform that task.

CALL and RET instruction: The **CALL** instruction transfer control of program to a new address specified in the instruction

- Every procedure/sub routine ends with RET instructions, thus the program control return back to main program

Instruction	Explanation
➔ CALL	Transfer control to another memory location either in same program or different program
➔ RET	Return from near call
➔ RETF	Return from far call

JUMP Instruction: They are of two types :-

i) Unconditional Jump Instruction : It does not check any flag condition

ii)Conditional Jump : Instruction is executed by checking flag condition

Unconditional Jump: Call-----Call a procedure and save return address on stack
RET------Return from procedure to calling program
JMP------Go to specified to get next instruction

Instruction	Explanation
JMP	Unconditional Jump
Conditional Jump Instructions :-	
JE	Perform jump if source operand = destination operand
JL	Jump if first operand is less that second operand

JLE	Perform jump if first operand is less that or equal to second operand
JB	Jump if first operand is below second operand
JBE	Jump if first operand is below or equal to second
JC	Jump if carry flag = 1
JNB	Jump is first operand is not below second
JNBE	Jump if First operand is not below or equal to second operand
JNL	Jump if first operand is not less than second
JNLE	Jump if first operand is not less that or not equal to second
JO	Jump if OF = 1
JNO	Jump if OF not equal to zero

Loop Instructions: Loop instructions are used to execute a group of instructions a number of times as specified by count value in CX Register

Instruction	Explanation
LOOP	Repeat the execution of group of instructions until the content of CX=0
LOOPZ / LOOPE	Repeat the execution of the group of instructions until CX=0 and ZF=1
LOOPNZ / LOOPNE	Repeat the execution of group of instruction until CX=0 and ZF=0

Processor Control Instructions: It includes the instructions to set or clear carry flag,direction flag and interrupt flag.

➔ **It also includes HLT,NOP,LOCK and ESC instructions**

Instructions	**Explanations**
CLC	**CF is reset to 0**
CMC	**Complement CF**
STC	**Set CF=1**
CLD	**Clear Direction flag**
STD	**Set DF=1**

CLI	**Interrupt flag is reset to 0**
STI	**Set IF = 1**
HLT	**Halt program execution**
NOP	**No operation is performed for 3 clock periods**
WAIT	**This isntruction allows the microprocessor to remain in wait state until a sign**