eg LDX AX [2000H]
LDX AX [BX]

4) LAHF :- This instruction load low order 8 bit of flag register into AH register

5) SAHF :- Stored the content of AH register to flag register

→ XLAT / XLATB :- This instruction read byte from look up table mainly used for code conversion. One code is placed in AL corresponding other code in lookup table

→ XCHG R/M :-

No flag affected

PUSH R/M :- This instruction push the contents of 16 bit register on to the top of stack

POP R/M :- This instruction pops from top of the stack and specified that content in 16 bit register

No flag affected

IN Port or IN AX or AL, DX :- This instruction read 8 or 16 bit data from code whose address in DX register

OUT Port :-

Airthematic Register :-

(1) ADD R,R , ADD R,M , ADD MR :- ADD AX [2000H]

:-

→ ADC RR , ADC RM , ADC MR

→ SUB RR , R/M , M/R . or SBB RR , R/M , M/R

Passion
Page No.
Date

Multiplication

flag affected overflow, carry flag will have chance of affect

MUL R/M :- This instruction multiply two unsigned no if content of specified register or memory location is 8 bit it will be multiplied by AL register and register stored in AL. If 16 bit the AX register.

IMUL R/M :- I → signed no multiplication

In case of Divide D, V is used

DIV R/M

IDIV R/M

Increment/Decrement :- flag affected (overflow, sign flag, zero, parity)

INC R/M

DEC R/M

DAA :- It is used after AND instruction to get correct result in BCD. This instruction for 8 bit operation. Sign flag, zero flag, carry flag, Auxillary flag & overflow flag.

DAS :- Decimal Adjust After Subtraction :- This instruction is used to get result in BCD when BCD no is subtracted from another BCD no and used for 8 bit operation.

(Conversion into Binary)

AAS / ASCII Adjust for Addition :- Numeric data which come from an input device into an microprocessor are usually a ASCII code 8086 allows addition of ASCII code of two decimal digit. This is used to get correct result in BCD format

Auxillary flag & carry flag will affected

AAD :- Adjust AX register for division. This instruction convert to BCD digit in AX into a Binary format. This is done by dividing two BCD digit in AX register by an another BCD byte

AAM :- Adjust result of BCD multiplication. Used for multiplication of two BCD no to get correct result & the finally Binary result is placed in AL register

Sign flag, Zero flag, Parity flag will get affected

No flag will affd

CBW :- Convert Byte Word :- This instruction copies Sign bit of an operand in AL register to all bit of AH register This is used before signed operation in AL and also known as Conversion of signed byte to signed word.

→ CWD :- Convert Word 32 bit Double word :- No flag will after

→ NEG M/R :- We can obtain two's complement from this instruction

→ Compare instruction :- All flag will effected (6 flag will affected)

CMP R/M
CMP data
CMP M,R
CMP M, data

Logical instruction :- ① AND R/M ② OR R/M ③ XOR R/M
AND data OR data XOR data
AND M,R OR M,R XOR M,R
AND M, data OR M, data XOR M data

Parity, carry, Signe, zero flag

→ NOT M/R :- This instruction used for 1's complement operation

TEST instruction :- Related ~~in~~ with ~~anding~~ AND operation but result will not be stored only required flag affected depending upon the operation.

→ Rotate instruction

- Rotate data with carry (RCL R/M, count) :- Rotate left through carry depending upon count operation. Count may be 1 depending upon the contents may be present in memory.

RCL AX, count=1

RCL AX [2000]

RCR :- RCR AX, count=1
RCR AX [2000H]

Rotation without carry (ROL R/M, count) :- This instruct rotate all bit of operand contain 8 bit or 16 bit register or memory location left by specific no of bits the MSB of operands moves to LSB position of the operand as well as to the carry flag

ROR :- ROR AX, count=1

→ Shift operation
i) SAL/SHL M/R, count :- Shift each bit of operand by specified no of bits that is shifting

ii) SAR/SHR M/R, count :- Max. 16 bit operation.

Branch instructions :-
JMP Jump instruction

JA :- Jump if Above JA 8 bit displacement

JNBE 8 bit displacement

JAE :- Jump if A more or may be equal to 8 bit displacement

JNB :- Jump if not Below 8 bit displacement

JNC :- Jump if no carry 8 bit displacement

Page No.
Date

JB : Jump if Below when carry flag = 1 — 8 bit displac

JNAE : Jump if not above & equal & carry flag = 1 — "

JC : Jump if carry — "

JBE : Jump if below & equal if zero flag = 1 — "

JNA : Jump if Not above but but zero flg = 1 — "

JNE

JE : Jump if Equal if zero flag = 1 — "

JZ — Jump if zero when zero flag = 1 — "

JG — Jump if greater — "

JMP 16 bit address or JMP 8 or 16 bit displacement
JMP M

→ CALL Instruction

CALL M/R/address/8 or 16 bit displacement (unconditional)

RET M/R/address/8 or 16 bit displacement

IRET : Return for interrupt service subroutine.

If External IC interrupt the processor then this instruction will be valid.

Interrupt instructions :-

1) INT n :- n stand from 0 to 256

INTO :- This interrupt will occur on interrupt on overflow.

loop Instruction

→ loop displacement :- JUMP to Specificed label until CX register is zero

LOOP E / LOOP Z Decrement CX register and Jump if CX is not equal to 0 & ZF = 1

LOOP LZ / LOOP LE :-

Flag Manuiplation instruction :-

1) CLC :-

CLD :- Direction flag

CLI :- Interrup

CMC :- Complement Carry flag

STC :- Set Carry flag

STD :- Set Direct

STI

HLT → Stop

NOP → No operation

ESC → Escape This instruction make bus free from external pheripheral device

Passion
Page No.
Date

IDEAL

WAIT :- When executed processor enter in ~~idle~~ state until test bar pin is made low

LOCK :- It is related with

String instruction.
(Related with Extra segment)

MOVS / MOVSW / MOVSB :- Move M to M.

CMPS / CMPSW / CMPSB :- Compare M to M.

SCAS / SCASW / SCASB :- Scan Memory to Memory & Compare contents ~~of~~ of accumulator with Content of memory strings to be scanned by ~~to~~ extra segment register

LODS, LODSB, LODSW :- Loading M to M
8 or 16 bit data from memory address by AX register

STOS / STOSB / STOSW) :- Store content AX register to IX register

→ Repeat Instruction

REP :- This instruction repeat strings until CX=0

REPE :- Repeat if Equal repeat if CX ≠ 0 & ZF=1

REPNE :- If CX ≠ 0 & ZF = 0