SNS COLLEGE OF ENGINEERING

(Autonomous)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

19SB405 – MICROPROCESSORS AND ADVANCED MICROCONTROLLERS

Instruction Groups

The 8086 has 117 instructions

The instructions are grouped into 5 groups

- Arithmetic
- Logic
- Data Transfer
- Bit Manipulation
- String Manipulation
- Program Execution transfer- Branching
- Process Control Execution

1. Arithmetic And Logical Instructions

Arithmetic Instructions

- **ADD Des, Src:**
  - It adds a byte to byte or a word to word.
  - It effects AF, CF, OF, PF, SF, ZF flags.
  - E.g.:
    - ADD AL, 74 H
    - ADD DX, AX
    - ADD AX, [BX]

Arithmetic Instructions

- **ADC Des, Src:**
  - It adds the two operands with CF.
  - It effects AF, CF, OF, PF, SF, ZF flags.
  - E.g.:
    - ADC AL, 74H
    - ADC DX, AX
    - ADC AX, [BX]

Arithmetic Instructions

- **SUB Des, Src:**
  - It subtracts a byte from byte or a word from word.
  - It effects AF, CF, OF, PF, SF, ZF flags.
  - For subtraction, CF acts as borrow flag.
  - E.g.:
    - SUB AL, 74H = (AL – 74)
    - SUB DX, AX
    - SUB AX, [BX]

Arithmetic Instructions

- **SBB Des, Src:**
  - It subtracts the two operands and also the borrow from the result.
  - It effects AF, CF, OF, PF, SF, ZF flags.
  - E.g.:
    - SBB AL, 74H (DES – SOURCE – CF)
    - SBB DX, AX
    - SBB AX, [BX]

Arithmetic Instructions

- **MUL Src:**
  - It is an unsigned multiplication instruction. MUL CX
  - It multiplies two bytes to produce a word or two words to produce a double word.
  - **MUL CX**
  - AX = AX * Src
  - DX : AX = AX * Src
  - This instruction assumes one of the operand in AL or AX.
  - Src can be a register or memory location. And Flags-OF, CF
  - Unused bits of destination register is always filled with sign bit

Arithmetic Instructions

- **DIV Src:**
- **DIV CX**
  - **= CX / AX**
  - It is an unsigned division instruction.
  - It divides word by byte or double word by word.
  - The operand is stored in AX, divisor is Src and the result is stored as:
    - AH = remainder, AL = quotient (for word/byte)
    - DX=remainder, AX=quotient (for D-word/word)
- **IDIV Src:**
  - It is a signed division instruction.

Arithmetic Instructions

- **CBW (Convert Byte to Word):**
  - This instruction converts byte in AL to word in AX.
  - The conversion is done by extending the sign bit of AL throughout AH
- **CWD (Convert Word to Double Word):**
  - This instruction converts word in AX to double word in DX : AX.
  - The conversion is done by extending the sign bit of AX throughout DX.

Arithmetic Instructions

- **INC Src:**
  - It increments the byte or word by one.
  - The operand can be a register or memory location.
  - E.g.: INC AX
- **DEC Src:**
  - It decrements the byte or word by one.
  - The operand can be a register or memory location.
  - E.g.: DEC AX

Arithmetic Instructions

- **CMP Des, Src:**
- **CMP AL, 74H (DES – SRC)**
  - It compares two specified bytes or words.
  - The Src and Des can be a constant, register or memory location.
  - Both operands cannot be a memory location at the same time.
  - The comparison is done simply by internally subtracting the source from destination.
- **NEG DES: 2’S COMPLEMENT**
  - It creates 2’s complement of a given number.
  - That means, it changes the sign of a number.

Arithmetic Instructions

- **DAA (Decimal Adjust after Addition)**
  - It is used to make sure that the result of adding two BCD numbers is adjusted to be a correct BCD number.
  - It only works on AL register.
- **For Subtraction : DAS (Decimal Adjust after Subtraction)**
- **15**
- **0001 0101**
- **11   A2**
- **1A   35**
- **2B + 06   D7 +60**

Logical Instructions

- **NOT Src:**
  - It complements each bit of Src to produce 1's complement of the specified operand.
  - The operand can be a register or memory location.
  - e,.g NOT AX
  - 110011100101
  - 001100011010

Logical Instructions

- **AND Des, Src: =0**
  - It performs AND operation of Des and Src.
  - Src can be immediate number, register or memory location.
  - Des can be register or memory location.
  - Both operands cannot be memory locations at the same time.
  - CF and OF become zero after the operation.
  - PF, SF and ZF are updated.
  - AND AX, 1240H

A	B	AND	OR	XOR
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

Bit Manipulation Instructions

- **OR Des, Src: OR AX , BX**
  - It performs OR operation of Des and Src.
  - Src can be immediate number, register or memory location.
  - Des can be register or memory location.
  - Both operands cannot be memory locations at the same time.
  - CF and OF become zero after the operation.
  - PF, SF and ZF are updated.

Bit Manipulation Instructions

- **XOR Des, Src:**
  - It performs XOR operation of Des and Src.
  - Src can be immediate number, register or memory location.
  - Des can be register or memory location.
  - Both operands cannot be memory locations at the same time.
  - CF and OF become zero after the operation.
  - PF, SF and ZF are updated.

Bit Manipulation Instructions

- **TEST Des, Src:**
  - It performs AND operation of Des and Src.
  - Src can be immediate number, and src/Des can be register or memory location.
  - It is Non-Destructive And means Dest is not modified only flags are affected.
  - Both operands cannot be memory locations at the same time.
  - CF and OF become zero after the operation.
  - PF, SF and ZF are updated.

2. Data Transfer Instructions

Data Transfer Instructions

- **MOV Des, Src:**
  - It is used to copy the content of Src to Des
  - Src operand can be register, memory location or immediate operand.
  - Des can be register or memory operand.
  - Both Src and Des cannot be memory location at the same time.
  - E.g.:
    - MOV CX, 037A H
    - MOV AL, BL
    - MOV BX, [0301 H]

Data Transfer Instructions

- **PUSH Operand:**
  - It pushes the operand into top of stack.
  - E.g.: PUSH BX

POP Des:

- It pops the operand from top of stack to Des.
- Des can be a general purpose register, segment register (except CS) or memory location.
- E.g.: POP AX

Data Transfer Instructions

- **XCHG Des, Src:**
  - This instruction exchanges Src with Des.
  - It cannot exchange two memory locations directly.
  - E.g.: XCHG DX, AX

Data Transfer Instructions

- **IN Accumulator, Port Address:**
  - It transfers the operand from specified port to accumulator register.
  - E.g.: IN AX, 0028 H

- **OUT Port Address, Accumulator:**
  - It transfers the operand from accumulator to specified port.
  - E.g.: OUT 0028 H, AX

Data Transfer Instructions

- **LEA Register, Src:**
  - It loads a 16-bit register with the offset address of the data specified by the Src.
  - E.g.: LEA BX, [DI]
    - This instruction loads the contents of DI (offset) into the BX register.

Data Transfer Instructions

- **LDS Des, Src:** (Load Pointer using DS)
  - It loads 32-bit pointer from memory source to destination register and DS.
  - The word is placed in the destination register and the segment is placed in DS.
  - This instruction Copies the word at the lower memory address to the Des reg and the word at the higher address to the segment reg i.e. DS.
  - E.g.: LDS BX, [0301 H]

Data Transfer Instructions

- **LES Des, Src:** (Load Pointer using ES)
  - It loads 32-bit pointer from memory source to destination register and ES.
  - The Word is placed in the destination register and the segment is placed in ES.
  - This instruction is very similar to LDS except that it initializes ES instead of DS.
  - E.g.: LES BX, [0301 H]

Data Transfer Instructions

- **LAHF:**
  - It copies the lower byte of flag register to AH.
- **SAHF:**
  - It copies the contents of AH to lower byte of flag register.
- **USHF:**
  - Pushes flag register to top of stack.
- **OPF:**
  - Pops the stack top to flag register.

3. Branch/Program Execution Transfer Instructions

- These instructions cause change in the sequence of the execution of instruction.
- This change can be a conditional or sometimes unconditional.
- The conditions are represented by flags.

Branch Instructions

- **CALL Des:**
  - This instruction is used to call a subroutine or functior or procedure.
  - The address of next instruction after CALL is saved onto stack.

RET:

  - It returns the control from procedure to calling program.
  - Every CALL instruction should have a RET.

Branch Instructions

- **JMP Des:**
  - This instruction is used for unconditional jump from one place

    to another.

Jxx Des (Conditional Jump):

- All the conditional jumps follow some conditional statements or any instruction that affects the flag.

Conditional Jump Table

Mnemonic	Meaning	Jump Condition
JA	Jump if Above	CF = 0 and ZF = 0
JAE	Jump if Above or Equal	CF = 0
JB	Jump if Below	CF = 1
JBE	Jump if Below or Equal	CF = 1 or ZF = 1
JC	Jump if Carry	CF = 1
JE	Jump if Equal	ZF = 1
JNC	Jump if Not Carry	CF = 0
JNE	Jump if Not Equal	ZF = 0
JNZ	Jump if Not Zero	ZF = 0
JPE	Jump if Parity Even	PF = 1
JPO	Jump if Parity Odd	PF = 0
JZ	Jump if Zero	ZF = 1

Loop Instructions

- **Loop Des:**
  - This is a looping instruction.
  - The number of times looping is required is placed in the CX register.
  - With each iteration, the contents of CX are decremented.
  - ZF is checked whether to loop again or not.

Program Execution Transfer Instructions

- **INTO (Interrupt on overflow):**
- This instruction generates type 4 interrupt (i.e. interrupt for overflow) and causes the 8086 to do an indirect far call a procedure which is written by the user to handle the overflow condition.

IRET

To return the execution to the interrupted program

4. Machine Control Instructions

Machine Control Instructions

HLT (Halt) :- It causes the processor to enter in to the halt state. It can be stop by INTR,NMI or RESET pin

NOP (No Opration) :- It causes the processor to enter in to the wait state for 3 Clock cycles.

WAIT :- It causes the processor to enter in to the ideal state. Can be stop by TEST, INTR OR NMI pin

LOCK :- This instruction prevents other processors to take the control of shared resources. For e.g LOCK IN AL,80H

5. Flag Manipulation Instructions

Flag Manipulation Instructions

- **STC:**
  - It sets the carry flag to 1.
- **CLC:**
  - It clears the carry flag to 0.
- **CMC:**
  - It complements the carry flag.

Flag Manipulation Instructions

- **STD:**
  - It sets the direction flag to 1.
  - If it is set, string bytes are accessed from higher memory address to lower memory address.

- **CLD:**
  - It clears the direction flag to 0.
  - If it is reset, the string bytes are accessed from lower memory address to higher memory address.

Flag Manipulation Instructions

- **STI:**
  - It sets the Interrupt flag to 1.
- **CLI:**
  - It clears the Interrupt flag to 0.

6. Shift And Rotate Instructio s

Shift And Rotate Instructions

- **SHL/SAL Des, Count:**
  - It shift bits of byte or word left, by count.
  - It puts zero(s) in LSBs.
  - MSB is shifted into carry flag.
  - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
  - However, if the number of bits to be shifted is more than 1, then the count is put in CL register. And recent bit to the CF (Carry flag)

Shift And Rotate Instructions

- **SHR/SAR Des, Count:**
  - It shift bits of byte or word right, by count.
  - It puts zero(s)(for SHL) and Sign bit (for SAL) in MSBs.
  - LSB is shifted into carry flag.
  - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
  - However, if the number of bits to be shifted is more than 1, then the count is put in CL register. And recent bit to the CF (Carry flag)

Shift And Rotate Instructions

- **ROL Des, Count:**
  - It rotates bits of byte or word left, by count.
  - LSB is transferred to MSB and also to CF.
  - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
  - However, if the number of bits to be shifted is more than 1, then the count is put in CL register. And recent bit to the CF (Carry flag)

Shift And Rotate Instructions

- **ROR Des, Count:**
  - It rotates bits of byte or word right, by count.
  - MSB is transferred to LSB and also to CF.
  - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
  - However, if the number of bits to be shifted is more than 1, then the count is put in CL register. And recent bit to the CF (Carry flag)

Shift And Rotate Instructions

- **RCL Des, Count:**
  - It rotates bits of byte or word right, by count.
  - LSB to MSB then MSB is transferred to CF and CF to LSB.
  - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
  - However, if the number of bits to be shifted is more than 1, then the count is put in CL register. And recent bit to the CF (Carry flag)

Shift And Rotate Instructions

- **RCR Des, Count:**
  - It rotates bits of byte or word left, by count.
  - MSB to LSB then LSB is transferred to CF and CF to MSB.
  - If the number of bits desired to be shifted is 1, then the immediate number 1 can be written in Count.
  - However, if the number of bits to be shifted is more than 1, then the count is put in CL register. And recent bit to the CF (Carry flag)

7. String Manipulation Instructions

String Manipulation Instructions

- String in assembly language is just a sequentially stored bytes or words.
- There are very strong set of string instructions in 8086.
- By using these string instructions, the size of the program is considerably reduced.

String Manipulation Instructions

- String : Sequence of bytes or words
- 8086 instruction set includes instruction for string movement, comparison, scan, load store.
- REP instruction prefix : used to repeat execution of string instructions
- String instructions end with S or SB or SW. S represents string, SB string byte and SW string word.
- Offset or effective address of the source operand is stored in SI register and that of the destination operand is stored in DI register.

Depending on the status of DF, SI and DI registers are automatically updated.

DF = 0 ⇒ SI and DI are incremented by 1 for byte and 2 for word.

DF = 1 ⇒ SI and DI are decremented by 1 for byte and 2 for word.

String Manipulation Instructions

- **MOVS / MOVSB / MOVSW:**
  - It causes moving of byte or word from one string to another.
  - In this instruction, the source string is in Data Segment referred by DS:SI and destination string is in Extra Segment referred by ES:DI.
  - For e.g. movs str1,str2
  - Movsb
  - Movsw

String Manipulation Instructions

- **LODS / LODSB / LODSW:**
  - It causes TRANSFER of byte or word from one string to another.
  - In this instruction, the source string is in Data Segment referred by DS:SI transferred to Accumulator.
  - For e.g. lods string
  - lodsb
  - lodsw

String Manipulation Instructions

- **STOS / STOSB / STOSW:**

It causes TRANSFER of byte or word from one string to another.

In this instruction, the string is in Extra Segment referred by ES:DI transferred to Accumulator.

- For e.g. stos string
- stosb
- stosw

String Manipulation Instructions

- **CMPS Des, Src:**
  - It compares the string bytes or words.

- **SCAS String:**
  - It scans a string.
  - It compares the String with byte in AL or with word in AX.

String Manipulation Instructions

- **REP (Repeat):**
  - This is an instruction prefix.
  - It causes the repetition of the instruction until CX becomes zero.
  - E.g.: REP MOVSB
    - It copies byte by byte contents.
    - REP repeats the operation MOVSB until CX becomes zero.

String Manipulation Instructions

REP	
REPZ/ REPE (Repeat CMPS or SCAS until ZF = 0)	While CX ≠ 0 and ZF = 1, repeat execution o string instruction and (CX) ← (CX) – 1
PNZ/ REPNE epeat CMPS or SCAS u til ZF = 1)	While CX ≠ 0 and ZF = 0, repeat execution of string instruction and (CX) ← (CX) - 1

Example – 16-bit Addition

Add 1E44 H to 56CA H

```
CLR     C          ; Clear the CY flag
MOV     Al, 44H    ; The lower 8-bits of the 1st number
ADD     Al, CAH    ; The lower 8-bits of the 2nd number
MOV     R1, A      ; The result 0EH will be in R1. CY = 1.
MOV     A, 1EH     ; The upper 8-bits of the 1st number
ADC     A, 56H     ; The upper 8-bits of the 2nd number
MOV     R2, A      ; The result of the addition is 75H
```

The overall result: 750EH will be in R2:R1. CY = 0.

1. **Explain the instruction with example**

THANK YOU