Operation

- If the overflow flag is set, this instruction will cause the 8086 to do an indirect far call to a procedure you write to handle overflow condition. To do this call the 8086 will read a new value for IP from address 0010 H and CS from 0012 H.

3. IRET – Interrupt return

Mnemonic IRET **Flags** No flags are affected.

Algorithm

- POP from stack
- IP
- CS
- Flag register

Operation The IRET instruction is used at the end of interrupt service routine to return execution to the interrupted program.

The 8086 copies return address from stack into IP, and CS registers and stored value of flags back to flag register.

Note : The RET instruction does not copy flags from the stack back to the flag register

12.11 String Instructions Group

Q. Explain with example string instructions of 8086 microprocessor.

Table 12.11.1 : String instructions

REP	Repeat
REPE/REPZ	Repeat while equal/zero
REPNE/REPNZ	Repeat while not equal/not zero
MOVSB/MOVSW	Move byte or word string
CMPSB/CMPSW	Compare byte or word string
SCASB/CMPSW	Scan byte or word string
LODSB/LODSW	Load byte or word string
STOSB/STOSW	Store byte or word string

1. MOVS/MOVSB/MOVSW : Move string byte or string word

Mnemonic MOVSB/MOVSW **Flags** No flags are affected.

Algorithm

In case of byte :

- ES : [DI] = DS : [SI]
- If DF = 0 then

  SI = SI + 1 DI = DI + 1

  else

  SI = SI – 1 DI = DI – 1

In case of word :

- ES : [DI] = DS : [SI]
- If DF = 0 then,

  SI = SI + 2 DI = DI + 2

  Else

  SI = SI – 2 DI = DI – 2

Addr. Mode String addressing mode

Operation

- This instruction copies a byte or a word from a location in the data segment to a location in the extra segment.
- The offset of the source byte/word in the DS must be SI register.
- The offset of the destination in ES must be in DI register.
- After byte or word is moved, SI and DI are automatically adjusted to point to the next source and next destination.

Example

```
LEA SI, SOURCE_STRING   ; SI points to start of source string.
LEA DI, DEST_STRING     ; DI points to start of destination string.
CLD                     ; Clear direction flag to use incrementing
MOVSB                   ; MOV (DI) ← (SI), byte transfer
```

2. CMPS : Compare string byte or string words

Mnemonic CMPSB/CMPSW **Flags** All flags are affected.

Two alternate mnemonics for the same instruction

CMPB, (compare string byte)

CMPW, (compare string word)

Algorithm

For byte operation

DS : [SI] – ES : [DI]

Set flags according to result → OF, SF, ZF, AF, PF, CF.

If DF = 0 then SI = SI + 1 DI = DI + 1

else SI = SI – 1 DI = DI – 1

For word operation

DS : [SI] – ES : [DI] Set flags, according to result → OF, SF, ZF, AF, PF, CF.

If DF = 0 then SI = SI + 2 DI = DI + 2

else SI = SI – 2 DI = DI – 2

Addr. Mode String addressing mode

Operation

- This instruction is used to compare a byte in one string with a byte in another string or to compare a word in one string with another string.
- SI is used to hold the offset of a byte or word in the source string and DI is used to hold the offset of byte or word is another string.
- Comparison is done by subtracting the byte or word pointed by DI from byte or word pointed by SI.
- After comparison SI and DI will be automatically incremented or decremented according to direction flag to point to next element in the string.

Example

```
LEA SI, SRC      ; SI points to start of source string labeled SRC.
LEA DI, DEST     ; DI points to start of destination string labeled DEST
CLD              ; Clear DF (Incrementing is used).
CMPS SRC, DEST ; Compare.
```

3. SCAS/SCASB/SCASW

(Scan a string Byte or Word)

Mnemonic SCASB/SCASW

SCASB (for byte operation)

Or SCASW (for word operation)

Flags No flags are affected.

Algorithm

For SCASB

- AL – ES : [DI]
- Set flags according to result
- If DF = 0 then DI = DI + 1

For SCASW

- AX – ES : [DI]
- Set flags according to result
- If DF = 0 then DI = DI + 2

  Else DI = DI – 2

Addr. Mode String addressing mode

Operation

- This instruction compares a byte in AL or word in AX with a byte or word pointed to by DI in ES. Hence the string to be scanned must be in ES, and offset in DI.
- After comparison DI will be automatically incremented or decremented according to direction flag DF.

Example Let BUFF ← 0EH
LEA DI, BUFF ; Load offset Buffer in DI
MOV AL, 0DH ; AL is loaded with the Byte to be compared. i.e. 0DH
; it is ASCII for carriage return it indicates line
; termination.
CLD ; Clear direction flag DF to use incrementing
SCASB ; Scan for byte in the string.

4. LODS

(Load string byte in AL or string word into AX.)

Mnemonic LODSB/LODSW **Flags** No flags are affected.
Or LODSB (For byte operation)
Or LODSW (For word operation)

Algorithm LODSB
- AL = DS : [SI]

If DF = 0 then SI = SI + 1 else SI = SI – 1

LODSW
- AX = DS : [SI]

If DF = 0 then SI = SI + 2 else SI = SI – 2

Addr. Mode String addressing mode

Operation This instruction copies a byte from a string location pointed to by SI to AX.

Example
CLD
MOV SI, OFFSET A1 : SI points to start of A1.
LODSB : Load byte in AL from the string

5. STOS/STOSB/STOSW

(Store byte or word in string)

Mnemonic STOSB/STOSW **Flags** No flags are affected
STOSB (For byte operation)
or STOSW (for word operation)

Algorithm For STOSB
- ES : [DI] = AL

If DF = 0 then DI = DI + 1 else DI = DI – 1

For STOSW
- ES : [DI] = AX

If DF = 0 then DI = DI + 2 else DI = DI – 2

Addr. Mode String addressing mode

Operation It transfers (copies) a byte or word from register AL or AX to the string element addressed in ES by DI. Depending on DF flag, DI is automatically incremented or decremented.

Example

```
MOV DI, OFFSET STR 1    ; DI points to top of string STR 1.
CLD                     ; DF = 0
MOV AX, 00H
STOSW                   ; Store string by AX value
```

6. REP/REPE/REPNE/REPNZ : (Prefix)

(Repeat string instruction until specified conditions exist).

Mnemonic REP/REPE/REPNE/REPNZ **Flags** –

Algorithm

Check_CX For REP
- CX = CX – 1
- Repeat the instruction to which it is a prefix
- go back to check _ CX

else
- exit from REP cycle

Operation
- REP is a prefix written before one of the string instructions. These instructions repeat until specified condition exists.

Instruction code	Condition for exit
REP	CX = 0
REPE/REPZ	CX = 0 or ZF = 0
REPNE/REPNZ	CX = 0 or ZF = 1

A. REPE/REPZ (Repeat while equal / Repeat while zero).

Mnemonic REPE/REPZ

Algorithm

```
Check CX
   if CX < > 0 then
        CX = CX – 1
        if ZF = 1 then
repeat the instruction to which it is prefix
go back to check CX
   else
   exit from REPE cycle
   else
exit from REPE cycle.
```

Operation
- These are two mnemonics for the same instruction.
- This prefix will cause the string instructions to be repeated as long as ZF = 1 and CX ≠ 0.
  If ZF = 0 or CX = 0, string instructions will not be repeated.

Example

```
LEA DI, STR 1     ; DI = offset of STR 1
MOV AL, 05 H      ; AL = 05 H
CLD               ; DF = 0
MOV CX, 35 H      ; CX = 35 H i.e. counter = 35 H
REPE SCASB        ; Repeat, scan string byte operation till
                  ; CX ≠ 0 and ZF = 1.
```

B. REPNE/REPNZ (Repeat while not equal / Repeat while not zero)

Mnemonic REPNE/REPNZ

Algorithm

Check CX
if CX < > 0 then CX = CX – 1
If ZF = 0 then
repeat the instruction to which it is prefix
go back to check CX
else exit from REPNE cycle
else exit from REPNE cycle.

Operation

- These are two mnemonics for the same instruction.
- This will cause string instructions to be repeated as long as ZF = 0 and CX ≠ 0.
- If ZF = 1 or CX = 0, string instructions will not be repeated.

❑❑❑