Type 5) **String Instructions of 8086** *(Very Important ※ **10m**)*

A **String** is a **series of bytes** stored sequentially in the memory. String Instructions operate on such "Strings".
The **Source String is at a location pointed by SI in the Data Segment.**
The **Destination String is at a location pointed by DI in the Extra Segment.**
The Count for String operations is always given by CX.
Since CX is a 16-bit register we can transfer max 64 KB using a string instruction.
SI and/or DI are **incremented/decremented** after each operation depending upon the direction flag **"DF"** in the flag register.
If **DF = 0**, it is **auto increment**. This is done by **CLD instruction**.
If **DF = 1**, it is **auto decrement**. This is done by **STD instruction**.

1) MOVS: MOVSB/MOVSW (*Move String*)

It is used to **transfer** a word/byte **from data segment to extra segment**.
The offset of the source in data segment is in SI.
The offset of the destination in extra segment is in DI.
SI and DI are incremented / decremented depending upon the direction flag.

Eg: **MOVSB** ; *ES:[DI] ← DS:[SI] … byte transfer*
; *SI ← SI ± 1 … depending upon DF*
; *DI ← DI ± 1 … depending upon DF*

MOVSW ; *{ES:[DI], ES:[DI + 1]} ← {DS:[SI], DS:[SI + 1]}*
; *SI ← SI ± 2*
; *DI ← DI ± 2*

2) LODS: LODSB/LODSW (*Load String*)

It is used to **Load AL** (or AX) register with a byte (or word) **from data segment**.
The offset of the source in data segment is in SI.
SI is incremented / decremented depending upon the direction flag (DF).

Eg: **LODSB** ; *AL ← DS:[SI] … byte transfer*
; *SI ← SI ± 1 … depending upon DF*

LODSW ; *AL ← DS:[SI]; AH ← DS:[SI + 1]*
; *SI ← SI ± 2*

3)STOS: STOSB/STOSW (*Store String*)

It is used to **Store AL** (or AX) **into** a byte (or word) in the **extra segment**.
The offset of the source in extra segment is in DI.
DI is incremented / decremented depending upon the direction flag (DF).

Eg: **STOSB** ; *ES:[DI] ← AL … byte transfer*
; *DI ← DI ± 1 … depending upon DF*

STOSW ; *ES:[DI] ← AL; ES:[DI+1] ← AH … word transfer*
; *DI ← DI ± 2 … depending upon DF*

4)CMPS: CPMSB/CMPSW (*Compare String*)

It is used to **compare** a **byte** (or word) **in** the **data segment with** a **byte** (or word) **in** the **extra segment**.
The offset of the byte (or word) in data segment is in SI. The offset of the byte (or word) in extra segment is in DI.
SI and DI are incremented / decremented depending upon the direction flag.
Comparison is done by subtracting the byte (or word) from extra segment from the byte (or word) from Data segment.
The Flag bits are affected, but the result is not stored anywhere.

Eg :**CMPSB** ; *Compare DS:[SI] with ES:[DI] … byte operation*
; *SI ← SI ± 1 … depending upon DF*
; *DI ← DI ± 1 … depending upon DF*

CMPSW ; *Compare {DS:[SI], DS:[SI+1]}*
; *with {ES:[DI], ES:[DI+1]}*
; *SI ← SI ± 2 … depending upon DF*
; *DI ← DI ± 2 … depending upon DF*

5)SCAS: SCASB/SCASW (*Scan String*)

It is used to **compare** the contents of **AL** (or AX) **with** a **byte** (or word) **in** the **extra segment**.
The offset of the byte (or word) in extra segment is in DI.
DI is incremented / decremented depending upon the direction flag (DF). Comparison is done by subtracting a byte (or word) from extra segment from AL (or AX). The Flag bits are affected, but the result is not stored anywhere.

Eg:**SCASB** ; *Compare AL with ES:[DI] … byte operation*
; *DI ← DI ± 1 … depending upon DF*

SCASW ; *Compare {AX} with {ES:[DI], ES:[DI+1]}*
; *DI ← DI ± 1 … depending upon DF*

REP (*Repeat prefix used for string instructions*)

This is an **instruction prefix**, which can be used in string instructions.
It can be **used with string instructions only**.
It **causes** the **instruction** to be **repeated CX number** of times.
After each execution, the **SI** and **DI** registers are **incremented/decremented** based on the **DF** (Direction Flag) in the Flag register **and CX is decremented.**
i.e. **DF = 1; SI, DI decrements.** #Please refer Bharat Sir's Lecture Notes for this ...

Thus, it is important that before we use the REP instruction prefix the following steps must be carried out:
CX must be initialized to the Count value. If **auto decrementing** is required, **DF** must be **set using STD** instruction **else cleared** using **CLD** instruction.

```
EG:     MOV CX, 0023H
        CLD
   REP  MOVSB
```

The above section of a program will cause the following string operation
ES:[DI] ← DS:[SI], SI ← SI + 1, DI ← DI + 1, CX ← CX – 1
to be executed 23H times (as CX = 23H) in auto incrementing mode (as DF is cleared).

6) REPZ/REPE (*Repeat on Zero/Equal*)
It is a conditional repeat instruction prefix.It behaves the same as a REP instruction **provided** the Zero Flag is set (i.e. **ZF = 1**).It is used with CMPS instruction.

☺ **For doubts contact Bharat Sir on 98204 08217**

7) REPNZ/REPNE (*Repeat on No Zero/Not Equal*)
It is a conditional repeat instruction prefix.It behaves the same as a REP instruction **provided** the Zero Flag is reset (i.e. **ZF = 0**).It is used with SCAS instruction.

Please Note: 8086 instruction set has only 3 instruction prefixes :

1) **ESC** *(to identify 8087 instructions)*
2) **LOCK** *(to lock the system bus during an instruction)*
3) **REP** *(to repeatedly execute string instructions)*

For a question on instruction prefixes (asked repeatedly), explain the above in detail.