	the LSB is stored in CF		becomes 1
SAL	Same as SHL	-	-
JG, JGE, JL, JLE, JE	Jumps for signed operations. G for greater, L for less, E for equal	-	cmp ax,bx JG LBL1
JA, JAE, JB, JBE, JE	Jumps for unsigned operations. A for above, B for below and E for equal		cmp ax,bx JA LBL1

STRING OPERATIONS :

⇒ We should set ES to DS before using these instruction like this :-

```
MOV AX,@DATA
MOV DS,AX
MOV ES,AX
```

⇒ Almost everything here works with **SI,DI** where SI lives in DS and DI lives in ES

⇒ DF or Direction Flag determines the direction of string operations, 0 means auto-increment and 1 means auto-decrement

CMPSB	Compares the byte at address DS:SI with the byte at the address ES:DI	Changes the flags	mov SI, offset str1 mov DI, offset str2 CMPSB JNE LBL2 LBL1: INC SI INC DI CMPSB JE LBL1 LBL2:....
CMPSW	Compares the word starting at address DS:SI with the word starting at the address ES:DI	Changes the flags	mov SI, offset str1 mov DI, offset str2 CMPSW JNE LBL2 LBL1: INC SI INC DI CMPSW JE LBL1 LBL2:....

STD	Sets the direction flag to auto-decrement	DF:=1	-
CLD	Clears the dir. Flag to auto-increment	DF:=0	-
REPE Comp_Operation	Repeats a comparison operation for number of times stored in CX as long as comparison operation returns equal	CX should contain no. of times Increments SI and DI everytime	mov cx,20h REPE CMPSB
REPNE	Same as above as long as **not equal**	CX should contain no. of times Increments SI and DI everytime	mov cx,20h REPNE CMPSB
REP	Repeats a comparison operation for number of times stored in CX **unconditionally**	CX should contain no. of times Increments SI and DI everytime	mov cx,20h REP CMPSB
MOVSB	Copies the byte at [DS:SI] to [ES:DI]	[ES:DI] = [DS:SI]	mov SI,offset str1 mov DI, offset str2 Mov cx,20d REP MOVSB
MOVSW	Copies the word at [DS:SI] to [ES:DI]	[ES:DI] = [DS:SI] [ES:DI+1] = [DS:SI+1]	mov SI,offset str1 mov DI, offset str2 Mov cx,10d REP MOVSW
LODSB	Loads the byte at [DS:SI] to AL **CAUTION:** **REP LODSB is not allowed !**	AL=[DS:SI]	-
LODSW	Loads the word starting at [DS:SI] to AX **CAUTION:** **REP LODSW is not**	AL=[DS:SI] AH=[DS:SI+1]	

	allowed !		
STOSB	Stores AL in [ES:DI]	[ES:DI] = AL	
STOSW	Stores AX in [ES:DI] and [ES:DI+1]	[ES:DI] = AL [ES:DI+1] = AH	
INSB	Loads a byte from I/O address stored in DX and Stores it in ES address stored in DI then increments DI by 1	[ES:DI] = IN(DX) DI += 1	mov dx,34f8h INSB
INSW	Loads a byte from I/O address stored in DX and Stores it in ES address stored in DI then increments DI by 1	[ES:DI] = IN(DX) DI += 2	mov dx,34f8h INSW
OUTSB	Loads a byte from DS address stored in SI to I/O address stored in DX then increments SI by 1	OUT(DX) = [DS:SI] SI += 1	mov dx,34f8h OUTSB
OUTSW	Loads a word from DS address stored in SI to I/O address stored in DX then increments SI by 2	OUT(DX) = [DS:SI] SI += 2	mov dx,34f8h OUTSW

SCASB	Compares AL with [ES:DI]		mov al,41H
			mov cx,20 ;Search the first 20 bytes of DI for 'A' char
			REPNE SCASB

XLAT	When BX carries the offset of an ASCII Lookup table of hex numbers from 0 to F, then AX is translates to its corresponding ASCII	BX:= offset ASCII_TABLE AX:= BX[AX]	mov al,9 mov BX,offset ASC_TBL XLAT