Instructions

1) Data Transfer Instructions

- **MOV**

  MOV AX,BX
  MOV AL,CH
  MOV CX,2050H
  MOV BH,33H

- **XCHG**

  XCHG BH, CL
  XCHG AX, DX

- **PUSH and POP**

  PUSH BX POP BX
  PUSH DS POP DS
  Note: POP CS is illegal

- **LEA: Load Effective Address**

  - Loads effective address of operand into specified register
    LEA AX, data1
  Alternative Instruction:
  MOV AX, offset data1

2) Arithmetic Instructions

- **ADD:-** Adds byte+byte OR word+word
  ADC:-Adds byte+byte+CY OR Adds word+word+CY
  Flags affected: AF, CF, OF, PF, SF, ZF
  E.g.

ADD AL, 25H;	AL <- AL+25H
ADD BL, AH;	BL <- BL+AH
ADD DX, BX;	DX <- DX+BX
ADD CH, [2050H];	CH<- CH+(byte from 2050H)
ADC AL, DL;	AL<- AL+DL+CY
ADD data1, AL;	data1<- data1+AL

  ADD data1, data2 ; (illegal: two variables cannot be added directly)
- **SUB:-** Subtracts byte-byte OR word-word
  SBB:-Subtracts byte-byte-CY OR Adds word-word-CY
  Flags affected: AF, CF, OF, PF, SF, ZF
  E.g.

SUB CX, BX;	CX <- CX-BX
SBB DH, AL;	DH <- DH-AL-CY
SUB CX, 2356H;	CX <- CX-2356H
SBB DX, [3427H];	DX <- DX-(word from 3427H)-CY

- **INC & DEC**

  INC reg; reg<-reg+1
  DEC reg; reg<-reg-1
  Flags affected: AF, OF, PF, SF, ZF
  E.g.

INC AH	DEC BL
INC BX	DEC DX

- **MUL & IMUL**
  MUL: Multiplies an unsigned byte from source times an unsigned byte in AL register or an unsigned word from source times an unsigned word in AX.
  In byte multiplication, result is kept in AX and in word multiplication, result is kept in DX:AX.
  IMUL: Same as **MUL,** except it does for signed value.
  E.g.
  MUL BL; AX <- AL*BL
  MUL CX; DX:AX<- AX*CX
  IMUL AH; AX <- AL*AH
  Flags affected: CF, OF; AF, PF, SF and ZF are undefined
- **DIV & IDIV**
  DIV: Divides unsigned word by byte or unsigned double word by word.
  When word is divided by byte, word must be in AX register and divisor is given in instruction.
  After division, AL<- quotient, AH<- remainder
  When double word is divided by word, most significant word must be in DX register and least significant word must be in AX register. Divisor is given in instruction.
  After division, AX<- quotient, DX<- remainder
  IDIV: Same as **DIV,** except it does for signed value.
  E.g.
  DIV CX; DX:AX/CX
  IDIV BL; AX/BL
  Flags affected: all flags are undefined
- **NEG: Negate (2's complement)**
  E.g.
  NEG AL
  NEG BX
  Flags affected: AF, OF, PF, SF, CF
- **CMP: Compare Bytes or words**
  E.g.
  CMP CX,AX CMP AL,20H CMP BX, 2050H
  If CX=AX; CF=0, ZF=1, SF=0
  If CX>AX; CF=0, ZF=0, SF=0
  If CX<AX; CF=1, ZF=0, SF=1
  AF, OF, PF are affected according to the result

3) Logical Instructions

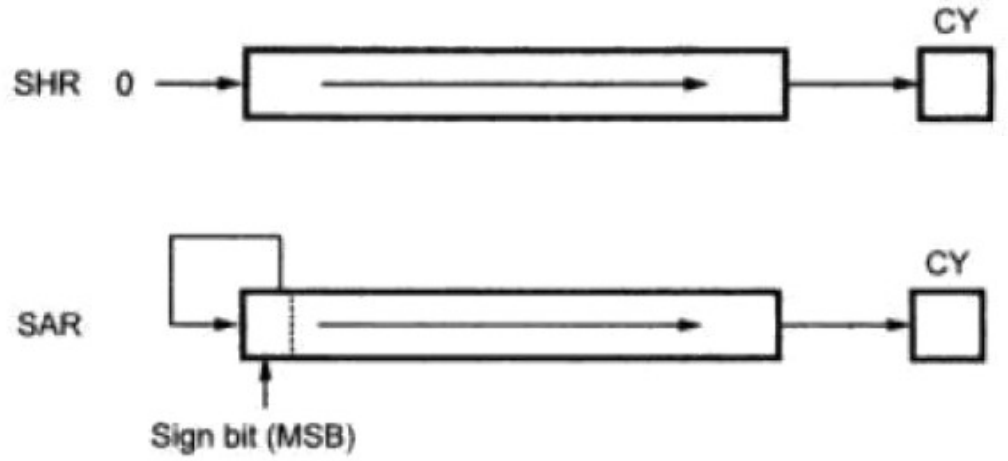
- **NOT – Flags:- No flags affected**
- **AND – Flags:- CF=0, OF=0, PF, SF, ZF according to the result**
- **OR – Flags:- CF=0, OF=0, PF, SF, ZF according to the result**
- **XOR – Flags:- CF=0, OF=0, PF, SF, ZF according to the result**
- **TEST:- AND operation to update flag, but neither operand is changed**

E.g.

NOT BX	AND BH,CL	AND BX,00ffH
OR CH,CL	OR BL,80H,	XOR BP, DI
XOR CL,0BH	TEST AL,BH	TEST CX,0010H

4) Shift Instructions

- **SHR/SAR**
  SHR (Shift Logical Right) -> for unsigned data
  SAR (Shift Arithmetic Right) -> for signed data
  Note: If the no. of shift is more than one, it should be loaded in CL register.



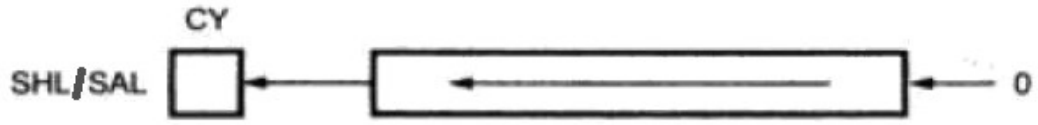
E.g.1

```
MOV BH, 10110111B
SHR BH, 01          ;BH=01011011, CY=1
MOV CL, 02
SHR BH, CL          ;BH=00101101, CY=1; 1st shift
                    ;BH=00010110, CY=1; 2nd shift
```

E.g.2

```
MOV BH, 00110111B
SAR BH, 01          BH=00011011,CY=1
MOV CL, 02
SAR BH, CL          BH=00001101, CY=1; 1st shift
                    BH=00000110, CY=1; 2nd shift
```

- **SHL/SAL**
  SHL (Shift Logical Left) -> for unsigned data
  SAL (Shift Arithmetic Left) -> for signed data
  (Both instructions do same operation)
  Note: If the no. of shift is more than one, it should be loaded in CL register.

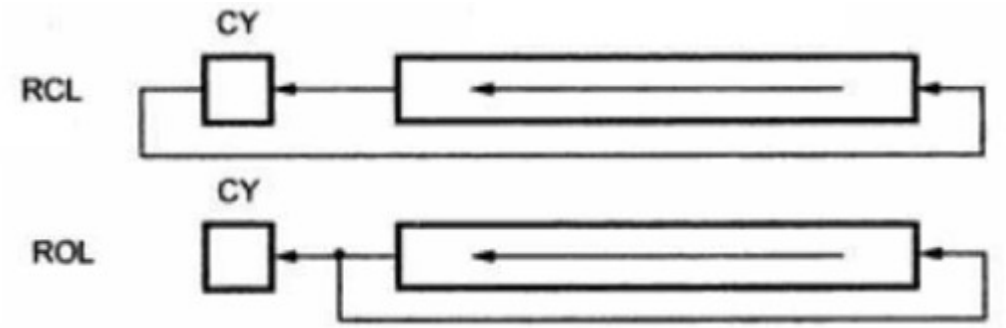


E.g.

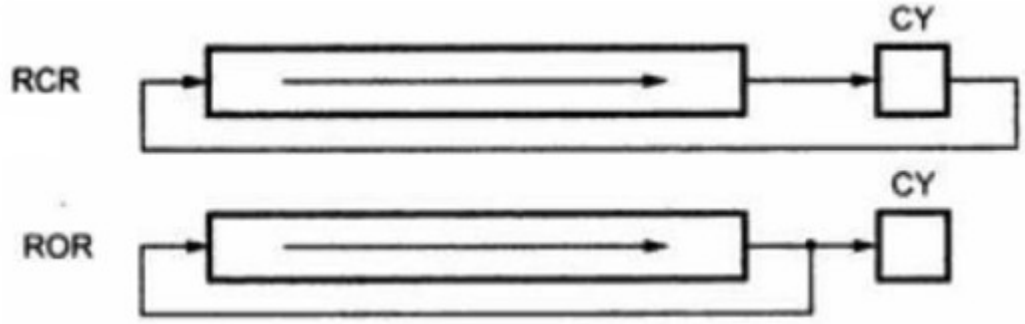
```
MOV BH, 00000101B
SHL BH, 01          BH=00001010, CY=0
MOV CL, 02
SAL BH, CL          BH=00010100, CY=0; 1st shift
                    BH=00101000, CY=0; 2nd shift
```

5) Rotate Instructions

- **ROL/RCL (Rotate left without carry/ Rotate left with carry)**



- **ROR/RCR (Rotate right without carry/ Rotate right with carry)**



E.g.1

```
MOV AL, 10100101B
ROL AL, 01      ;AL=01001011, CY=1
MOV CL, 02
ROL AL, CL      ;AL=10010110, CY=0; 1st rotate
                ;AL=00101101, CY=1; 2nd rotate
```

E.g.2

```
MOV AL, 10100101B
ROR AL, 01      ;AL=11010010, CY=1
MOV CL, 02
ROR AL, CL      ;AL=01101001, CY=0; 1st rotate
                ;AL=10110100, CY=1; 2nd rotate
```