8086 Instruction Set

Arithmetic Instruction

Arithmetic Instruction

- ADD
- ADC
- SUB
- SBB
- INC
- DEC
- CMP

Arithmetic Instruction

- MUL
- IMUL
- DIV
- IDIV
- NEG
- CBW
- CWD

MUL

- **MUL** ***source***(unsigned 8/16-bit register)
- If the source is *8-bit* ,it is multiplied with AL and **result is stored in AX**(AH-higher byte and AL lower byte.)
- If the source is *16-bit* ,it is multiplied with AX and **result is stored in DX-AX**(DX-higher byte and AX lower byte.)

MUL

Source : Register , Memory location

MUL affects **AF,PF,SF and ZF.**

Example: MUL BL

MUL BX

MUL BYTE PTR[BX]

IMUL

- **IMUL** ***source***(signed 8/16 bit register)
- Same as MUL except the source is a signed number

DIV

DIV ***source***(unsigned 8/16-bit register)

This instruction is used for unsigned division.

Divides a word by a byte or double word by a word.

If the divisor is 8-bit , the dividend is in AX register After division ,the **quotient is in AL and remainder in AH**

If the divisor is 16-bit , the dividend is in DX-AX register After division ,the **quotient is in AX and remainder in DX**

DIV

Source :Register , Memory location

All Flags are *undefined* after DIV instruction.

Example: DIV BL   // AX ÷ BL AL<- Quotient
                                        AH <-remainder

DIV BX // {DX,AX} ÷ BX
            AX<- Quotient
             DX <-remainder

IDIV

IDIV ***source*** {signed 8/16 bit register –divisor}

Same as DIV except that it is used for UNSIGNED division.

NEG

- **NEG** ***destination***

- This instruction forms the 2’s complement of the destination and stores result in destination.

- Destination: Register , Memory location

- ***All condition flags are updated***

NEG

- Example: AL= 0011 0101=35H
- NEG AL // AL= 1100 1011 =CBH

CBW

Convert signed byte to signed word

This instruction copies sign of the byte in AL into all the bits of AH .

AH is then called sign extension of AL.

No flags are affected.

Example: AX=XXXX XXXX 1001 0001
then CBW gives
AX= 1111 1111 1001 0001

CWD

Convert signed WORD to signed DOUBLE WORD

This instruction copies sign of the word in AX into all the bits of DX .

DX is then called sign extension of AX.

No flags are affected.

Example:AX= 1000 0000 1001 0001 DX=XXXX XXXX XXXX XXXX

then CWD gives

AX=1000 0000 1001 0001 DX= 1111 1111 1111 1111

Arithmetic Instruction

- Decimal Adjustment Instruction
  - DAA
  - DAS
- ASCII Adjustment Instruction
  - AAA
  - AAS
  - AAM
  - AAD

Decimal Adjustment Instruction

- DAA(Decimal Adjust for Addition)
- It makes the result in packed BCD form after BCD addition is performed.
- It works only on AL register.
- flags are updated; OF becomes undefined after this instruction.

DAA

- IF **D_3-D_0 >9** OR **Auxillary Carry Flag is set**=> **ADD 06H to AL.**
- IF **D_7-D_4 >9** OR **Carry Flag is set**=> **ADD 60H to AL.**

DAA

- Example:

  AL=14H

  CL=28H

  Then ADD AL,CL GIVES

  AL=3CH

  Now DAA gives

  AL=42 (06 is added to AL as C > 9)

8086 Instruction Set

Bit Manipulation Instruction

Bit Manipulation Instruction

Logical Instruction

Bit Manipulation Instruction

1.Logical Instruction

2.Shift Instruction

3.Rotate Instruction

NOT

NOT ***destination***

- Bit 1 is turn to 0 and bit 0 is turn to 1.
- No flags are affected.

i.e. find 1’s complement of number .

- Destination : register, memory location.
- ***Cannot be segment register***.

NOT

Assume

AX= 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 1

NOT AX

NOT AX= 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 0

AND

AND ***destination , source***

- This instruction logically ANDs the source with the destination and stores result in the destination.
- Source and destination have to be of the same size.
- Source :Register ,Memory location , Immediate data
- Destination: Register ,Memory location

AND

- After execution PF,SF,ZF affected ;
  CF=0 ;OF=0 and
  AF becomes **undefined**

- AND BL,CL   ;BL=BL AND CL

OR

OR ***destination ,source***

- This instruction logically ORs the source with the destination and stores result in the destination.

- Source and destination have to be of the same size.
- Source :Register ,Memory location , Immediate data
- Destination: Register ,Memory location

OR

- After execution PF,SF,ZF affected ;
  CF=0 ;OF=0 and
  AF becomes **undefined**

- OR AL,BL

XOR

XOR ***destination, source***

- This instruction logically X-ORs the source with the destination and stores result in the destination.
- Source and destination have to be of the same size.
- Source :Register ,Memory location , Immediate data
- Destination: Register ,Memory location

XOR

- After execution PF,SF,ZF affected ;
  CF=0 ;OF=0 and
  AF becomes **undefined**

- XOR AL,BL

TEST

TEST ***destination , source***

- This instruction logically ANDs the source with the destination BUT result is NOT stored anywhere.
- Used to set flags before conditional jump.
- TEST AL,75H

Bit Manipulation Instruction

Shift Instructions

- Up to 255 shift may be performed.
- Example:1.Arithmetic Shift

  2.Logical Shift

SAL/SHL

SAL/SHL ***destination , count***

- Left shifts the bits of destination.
- MSB shifted into the CARRY .
- LSB gets a 0.
- Bits are shifted ‘count’ number of times

SAL/SHL

- IF count=1 It is directly specified in the instruction.
- IF count >1 ,it has to be loaded in CL register

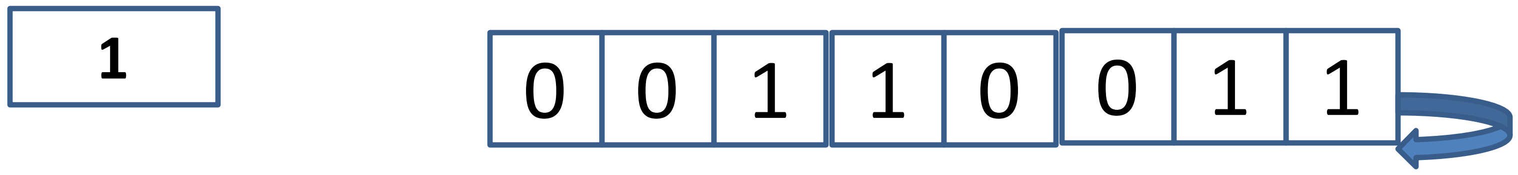
- Destination: Register, Memory Location

- SAL BL,1   // left shift bits once

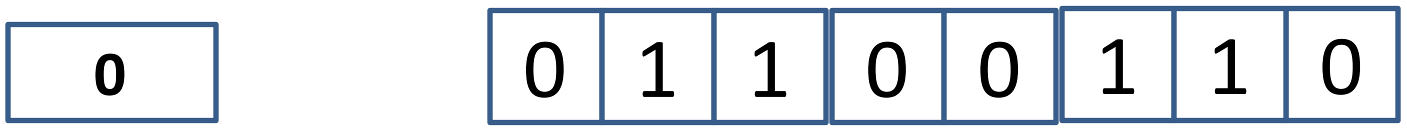
- Before operation BL= 0 0 1 1 0 0 1 1

COUNT=1

CARRY Destination



After Operation



SHR

SHR ***destination , count***

- Right shifts the bits of destination.
- LSB shifted into the CARRY .
- MSB gets a 0.
- Bits are shifted ‘count’ number of times

SHR

- IF count=1 It is directly specified in the instruction.
- IF count >1 ,it has to be loaded in CL register

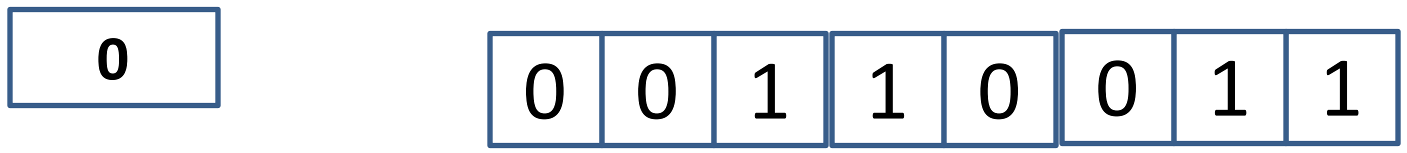
- Destination: Register, Memory Location

- SHR BL,1   // left shift bits once

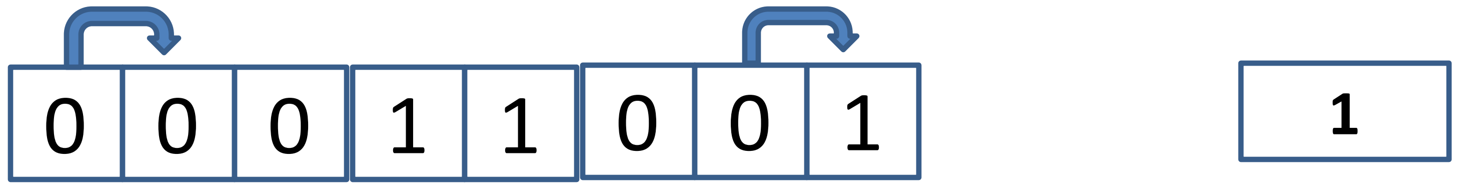
- Before operation BL= 0 0 1 1 0 0 1 1

COUNT=1

CARRY Destination



After Operation



SAR

SAR ***destination , count***

- Right shifts the bits of destination.
- LSB shifted into the CARRY .
- MSB PLACED IN MSB itself(Sign is preserved)
- Bits are shifted ‘count’ number of times

SAR

- IF count=1 It is directly specified in the instruction.
- IF count >1 ,it has to be loaded in CL register

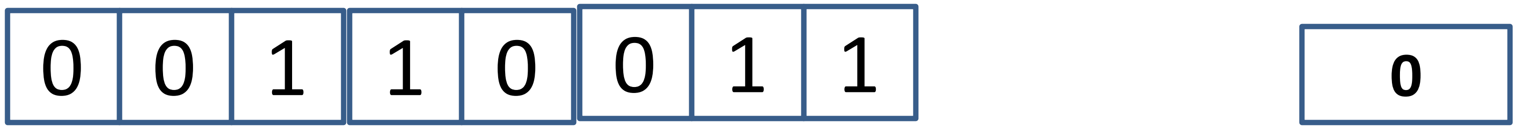
- Destination: Register, Memory Location

- SAR BL,1   // left shift bits once

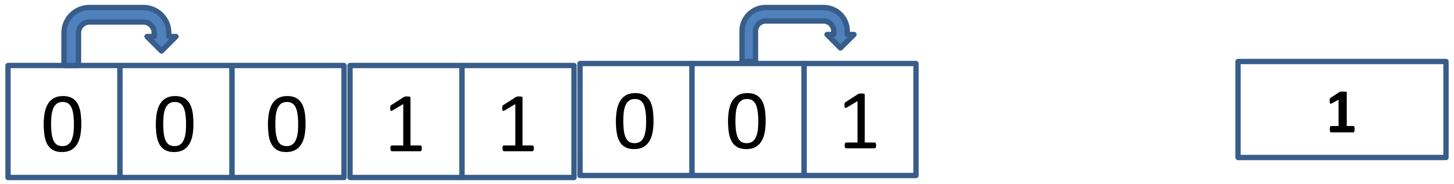
- Before operation BL= 0 0 1 1 0 0 1 1

COUNT=1

Destination CARRY



After Operation



8086 Instruction Set

Rotate Instruction

ROL

ROL: Rotate Left Byte/word

ROL destination, source

- Left shifts the bits of destination.
- MSB shifted into CARRY
- MSB also goes to LSB.

ROL

- Bits are shifted ‘count’ number of times

- IF count=1 It is directly specified in the instruction.
- IF count >1 ,it has to be loaded in CL register

- Destination: Register, Memory Location

- ROL BL,1   // left shift bits once

ROL

Carry   Destination

$$\begin{tikzcd}
CF & D7 \arrow[l] \arrow[rrrrrrr, bend right] & D6 & D5 & D4 & D3 & D2 & D1 & D0
\end{tikzcd}$$

ROR

ROR: Rotate Right Byte/word

ROR destination, source

- Right shifts the bits of destination.
- LSB shifted into CARRY
- LSB also goes to MSB.

RCL

RCL:

RCL destination, source

- Left shifts the bits of destination.
- MSB shifted into CARRY
- CF also goes to LSB

RCR

RCR:

RCR destination, source

- Right shifts the bits of destination.
- LSB shifted into CARRY
- CF also goes to MSB

8086 Instruction Set

Process Control/Machine Control Instruction

CARRY FLAGS

1. STC

- This instruction sets the carry flag.
- No other flags are affected.

2. CLC

- This instruction clears the carry flag.
- No other flags are affected.

3. CMC

- This instruction Complements the carry flag.
- No other flags are affected.

DIRECTION FLAG

1.STD

- This instruction sets the direction flag.
- No other flags are affected.

2.CLD

- This instruction clears the direction flag.
- No other flags are affected.

Interrupt Enable Flag

1.STI

- This instruction sets the Interrupt Enable flag.
- No other flags are affected.

2.CLI

- This instruction clears the Interrupt Enable flag.
- No other flags are affected.

Trap Flag

- To set TRAP Flag

```
PUSH     //push the content of flag register to stack
POP BX  //pop the contents  of flag register from
               top of stack to BX
OR BH,01H // set the bits corresponds to TF  in
                  the BH register
PUSH BX  //push the modified BX register on stack
POPF      //Pop the modified content into flag
                register
```