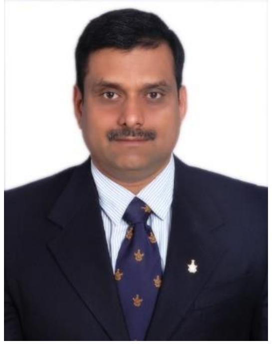
Microprocessors and Interfaces: 2021-22
Lecture 13
8086 Arithmetic Instructions : Part-2

By Dr. Sanjay Vidhyadharan

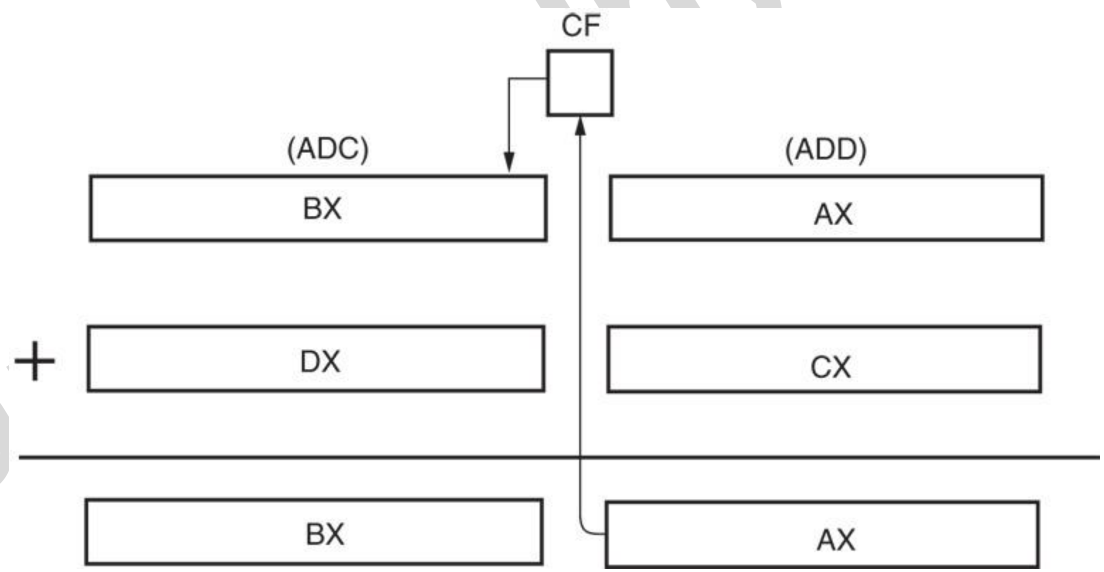


Addition

- **Addition : ADD, ADC**
- **ADD Destination, Source**

  (Source) + (Destination)   (Destination)
- **ADC   DESTINATION, SOURCE**

  (SOURCE) + (DESTINATION) + CF   (DESTINATION)

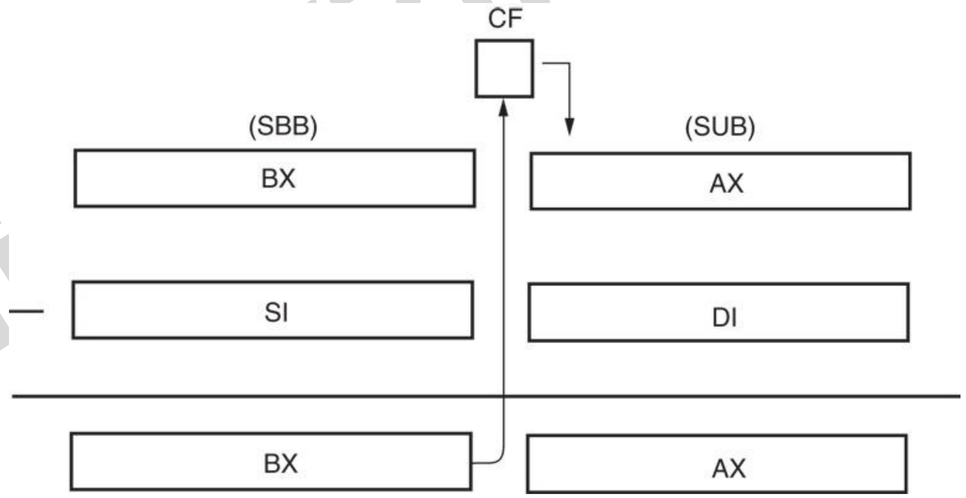


Subtraction

- **Subtraction, SUB , SBB**

SUB DESINATION, SOURCE
(DESTINATION) – (SOURCE) (DESTINATION)

SBB DESTINATION, SOURCE
(DESTINATION) - (SOURCE) – CF (DESTINATION)



Compare Instruction

- ✓ Compare instruction is a subtraction that changes only the flag bits. Destination operand never changes

- ✓ CMP Destination, Source

	CF	ZF	SF
Equal	0	1	0
dest > source	0	0	0
dest < source	1	0	1

Ex: CMP CL, [BX]
CMP AX, 2000H
CMP [DI], CH

8-Bit Multiplication

- With 8-bit multiplication, the multiplicand is always in the AL register, signed or unsigned.
  - multiplier can be any 8-bit register or memory location

- Immediate multiplication is not allowed unless the special signed immediate multiplication (in 80186) instruction appears in a program.

- Eg. MUL BL (Product in AX)
- Eg. MUL Byte PTR [BX] (Product in AX)

16-Bit Multiplication

- Word multiplication is very similar to byte multiplication.
- AX contains the multiplicand instead of AL.
  - 32-bit product appears in DX–AX instead of AX
- The DX register always contains the most significant 16 bits of the product; AX contains the least significant 16 bits.

Eg. MUL BX (Product in DX-AX)
Eg. MUL Word PTR [BX] (Product in DX-AX)

Division

- Occurs on 8- or 16-bit numbers.
  - signed (IDIV) or unsigned (DIV) integers
- Dividend is always a double-width dividend, divided by the operand.
- There is no immediate division instruction available to any microprocessor.

Eg . DIV BL (Contents of AX divided by BL
Quotient in AL and Remainder in AH)

Eg . DIV BH (Contents of DX-AX divided by BH
Quotient in AX and Remainder in DX)

Increment

- The INC instruction adds 1 to any register or memory location, except a segment register.
- The size of the data must be described by using the BYTE PTR, WORD PTR directives.
- The assembler program cannot determine if the INC [BX] instruction is a byte-, word-sized increment.

Ex: INC CX ; Add 1 to the contents of CX.
Ex: INC DI ; Add 1 to the contents of DI.
EX: INC BYTEPTR [DI] ; Increments the byte pointed to by the contents of DI.

(AF, OF, PF, SF, ZF affected, CF not affected)

INC/DEC the contents of a Memory location

- Specify the data size in memory

  use directive

- BYTE PTR, WORD PTR, DWORD PTR
- INC WORD PTR [BX]
- INC BYTE PTR[BX]
- BX-1000_H DS-2000_H

Consider

21000	**FF**
21001	00

After execution of INC WORD PTR [BX]

21000	**00**
21001	01

After execution of INC BYTE PTR [BX]

21000	**00**
21001	00

BCD and ASCII Arithmetic

- The microprocessor allows arithmetic manipulation of both BCD (**binary-coded decimal**) and ASCII (**American Standard Code for Information Interchange**) data.

BCD Arithmetic

- Two arithmetic techniques operate with BCD data:
- addition and subtraction.
- DAA (**decimal adjust after addition**) instruction follows BCD addition,
- DAS (**decimal adjust after subtraction**) follows BCD subtraction.
  - both correct the result of addition or subtraction so it is a BCD number

DAA

- DAA follows the ADD or ADC instruction to adjust the result into a BCD result.
- After adding the AL and BL registers, the result is adjusted with a DAA instruction before being stored.
- Ex: before execution let AL =0101 1001=59 BCD and

BL= 0011 0101= 35 BCD

```
ADD AL,BL      ;  AL =1000 1110= 8EH
DAA            ; Add 0110 because 1110 > 9
               ; AL= 1001 0100= 94 BCD
```

AF,CF,PF and ZF are affected. OF is undefined after DAA instruction.

DAS Instruction

- Functions as does DAA instruction, except it follows a subtraction instead of an addition.
- Ex: AL=1000 0110 =86 BCD

  BH= 0101 0111 =57 BCD

  SUB AL,BH ; AL= 0010 1111 =2FH,CF=0

  DAS ; lower nibble=1111>9 So,DAS subtracts 0000 0110 to give AL=0010 1001 =29 BCD

ASCII Arithmetic

- ASCII arithmetic instructions function with coded numbers, value 30H to 39H for 0–9.
- Four instructions in ASCII arithmetic operations:
  - AAA (ASCII adjust after addition)
  - AAD (ASCII adjust before division)
  - AAM (ASCII adjust after multiplication)
  - AAS (ASCII adjust after subtraction)
- These instructions use register AX as the source and as the destination.

AAA Instruction

- Addition of two one-digit ASCII-coded numbers will not result in any useful data.
- Ex: Before: AL= 0011 0001 , ASCII 1;

  BL= 0011 1001,ASCII 9

  ADD AL,BL ; Result : AL=0110 1110 = 6AH,

  ; which is incorrect ASCII

  AAA ;

  ADD AX, 3030

✓ The AAA instruction works only on the AL register.

✓ The AAA instruction updates AF and CF but OF,PF,SF and ZF are left undefined.

AAD(BCD to Binary convert before Division)

- Appears before a division.
- The AAD instruction requires the AX register contain a two-digit unpacked BCD number (not ASCII) before executing.
- Ex: AX= 0607H unpacked BCD for 67 decimal

```
CH=09 H , now adjust to binary
AAD        ; result: AX=0043=43H= 67 decimal
DIV CH     ; Divide AX by unpacked BCD in CH
           ; quotient :  AL=07 unpacked BCD
           ; Remainder : AH=04 unpacked BCD
           ; Flags  undefined after DIV
```

AAM (BCD Adjust after multiply)

- Follows multiplication instruction after multiplying two one-digit unpacked BCD numbers.
- AAM converts from binary to unpacked BCD.
- Ex: AL= 00000101 =unpacked BCD 5

```
    BH=00001001 = unpacked BCD 9
    MUL BH        ; AL X BH, result in AX
                  ; AX =00000000 00101101 =002DH
    AAM           ; AX=0000 0100 00000101= 0405H
                  ; which is unpacked BCD for 45.
```

; if ASCII codes for the result are desired, use next instruction.

ADD AX,3030H ; put 3 in upper nibble of each byte.
; AX=00110100 00110101 =3435H
; which is ASCII code for 45

AAS Instruction

- AAS adjusts the AX register after an ASCII subtraction.
- Ex1: AL=00111001 =39H =ASCII 9

  BL= 00110101 =35H= ASCII 5

  SUB AL,BL ;Result: AL= 00000100= BCD 04

  and CF=0

  AAS ; result: AL=00000100 = BCD 04

  and CF=0, no borrow required.

  ASCII 5 - ASCII 9(5-9)

Ex2: AL= 00110101 =35H

BL= 00111001 =39H

SUB AL,BL ; Result : AL= 11111100 = -4 in 2's

; complement and CF=1

AAS ; Result: AL=00000100 =BCD 04

; and CF=1, borrow needed

- ✓ The AAS instruction leaves the correct unpacked BCD result in the lower nibble of AL and resets the upper nibble of AL to all 0's

XADD Instruction

XADD dest, source   Exchange (content of operands) and add

XADD BL, CL

After execution both the operand content will change.

Thankyou