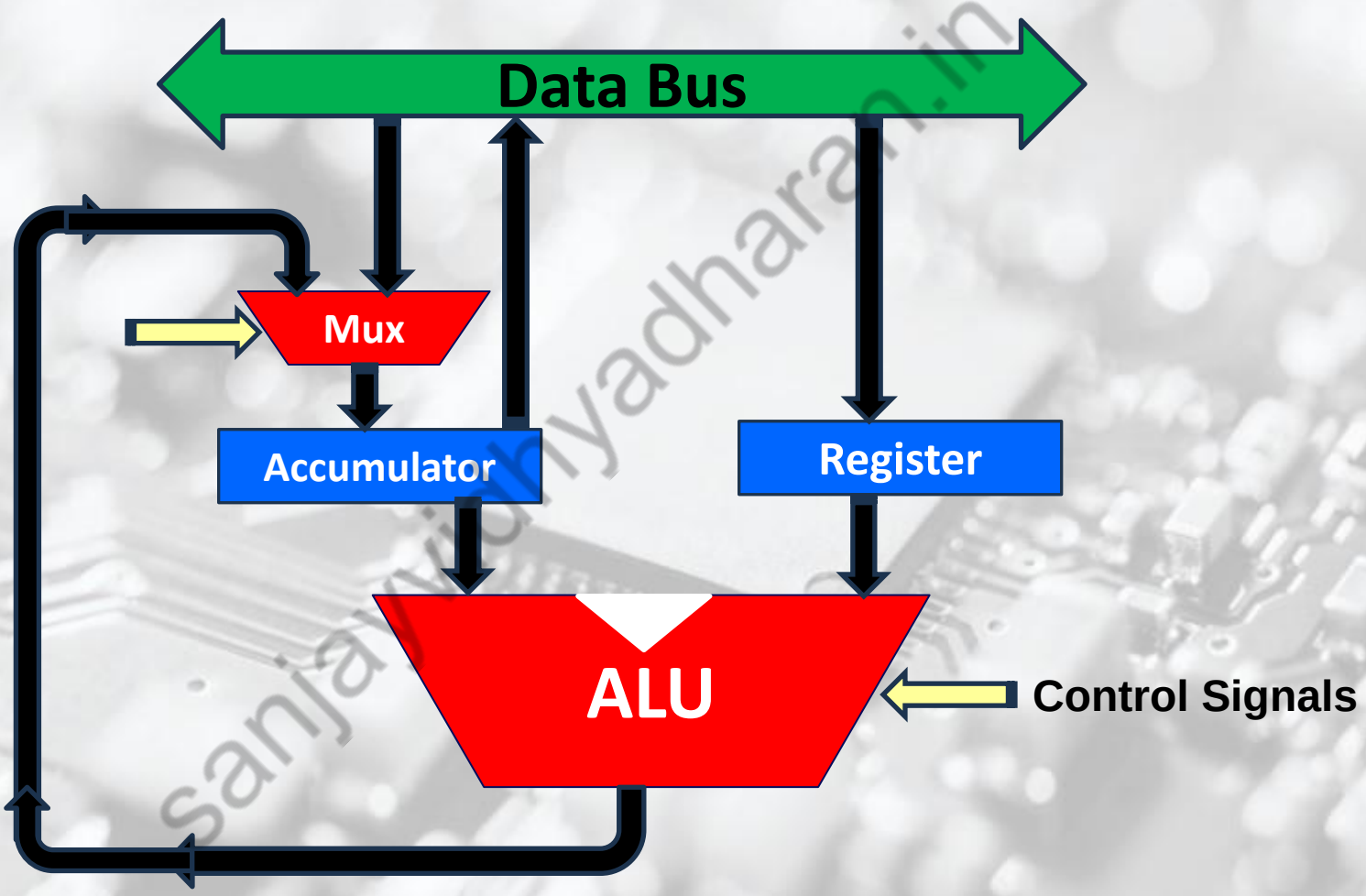
Microprocessors and Interfaces:
Lecture 12
8086 Arithmetic Instructions : Part-1

By Dr. Sanjay Vidhyadharan

sanjayvidhyadharan.in

Microprocessor ALU



sanjayvidhyadharan.in

Arithmetic Instructions

- Addition
- Subtraction
- Increment
- Decrement
- Comparison
- Multiplication
- Division

Flags get affected by Arithmetic Operations

sanjayvidhyadharan.in

Addition

add destination, source

Performs

destination := destination + source

add register, register

```
ADD AL,BL        AL = AL + BL
ADD CX, DI       CX = CX + DI
```

add register, immediate

```
ADD CL,44H       CL = CL + 44H
ADD BX,245FH     BX = BX + 245FH
```

add memory, immediate

```
ADD [CX],44H     [CX] = [CX] + 44H
```

add register, memory

```
ADD CL,[BP]
ADD BX,[SI+2]
```

add memory, register

There are several variations of the ADD instruction in the instruction set,.
The only types of addition not allowed are memory-to-memory and segment register.
The segment registers can only be moved, pushed, or popped.

sanjayvidhyadharan.in

Addition

Example 1:

Add data byte in memory with offset 1000h and 1001h and store result in memory with offset 1002h

```
MOV DI,1000h
MOV AL,0h
ADD AL,[DI]
ADD AL,[DI+1]
MOV [DI+2], AL
```

Addition is not commutative

sanjayvidhyadharan.in

Addition

Example 2:

Add word stored in memory with offset 1000h and 1002h and store result in 1004h

```
MOV AX,[1000h]
MOV BX,[1002h]
ADD AX,BX
MOV [1004h],AX
```

Example: 3

MOV DL, 13H

ADD DL, 32H

The sum 45H is stored in DL register.

Flags after the operation

Z = 0 (result not zero), S = 0 (result positive), C = 0 (no carry),

P = 0 (odd parity),

AC = 0 (no half carry), O = 0 (no overflow).

```
  0001  0011
+ 0011  0010
  0100  0101
```

More solved examples available in Tutorials

sanlayvidhyadharan.in

Addition

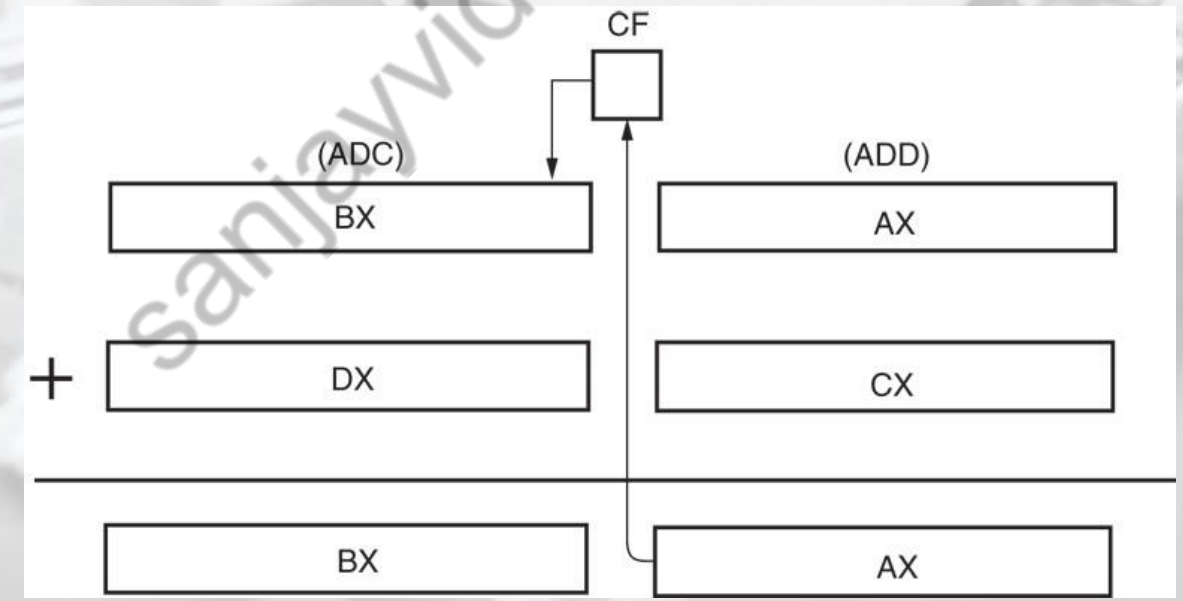
adc destination, source

Performs

destination := destination + source + CF

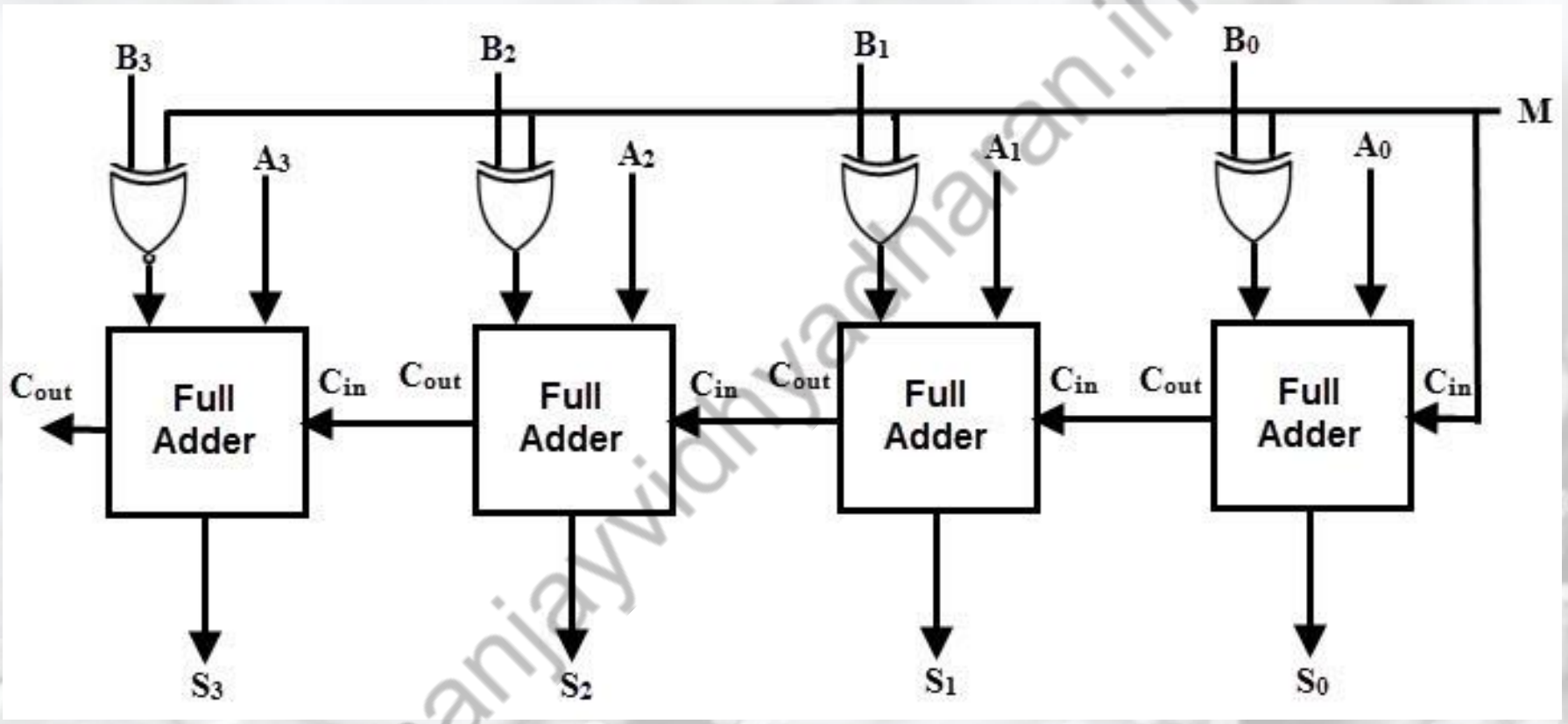
Carry flag can be manipulated by stc, clc, cmc

Addition-with-carry showing how the carry flag (C) links the two 16-bit additions into one 32-bit addition.



sanjayvidhyadharan.in

Subtraction



sanjayvidhyadharan.in

Subtraction

sub destination, source

Performs

destination := destination - source

Examples:
SUB AL, BL
SUB AX, CX
SUB AX, [DX]

sbb destination, source

Performs

destination := destination - source - CF

sanjayvidhyadharan.in

Subtraction

Example 1:

MOV AL, 22H

SUB AL, 44H gives

0010 0010

1011 1100

1101 1110

Flags after the operation

Z = 0 (result not zero), S = 1 (result negative), C = 0 (no carry),

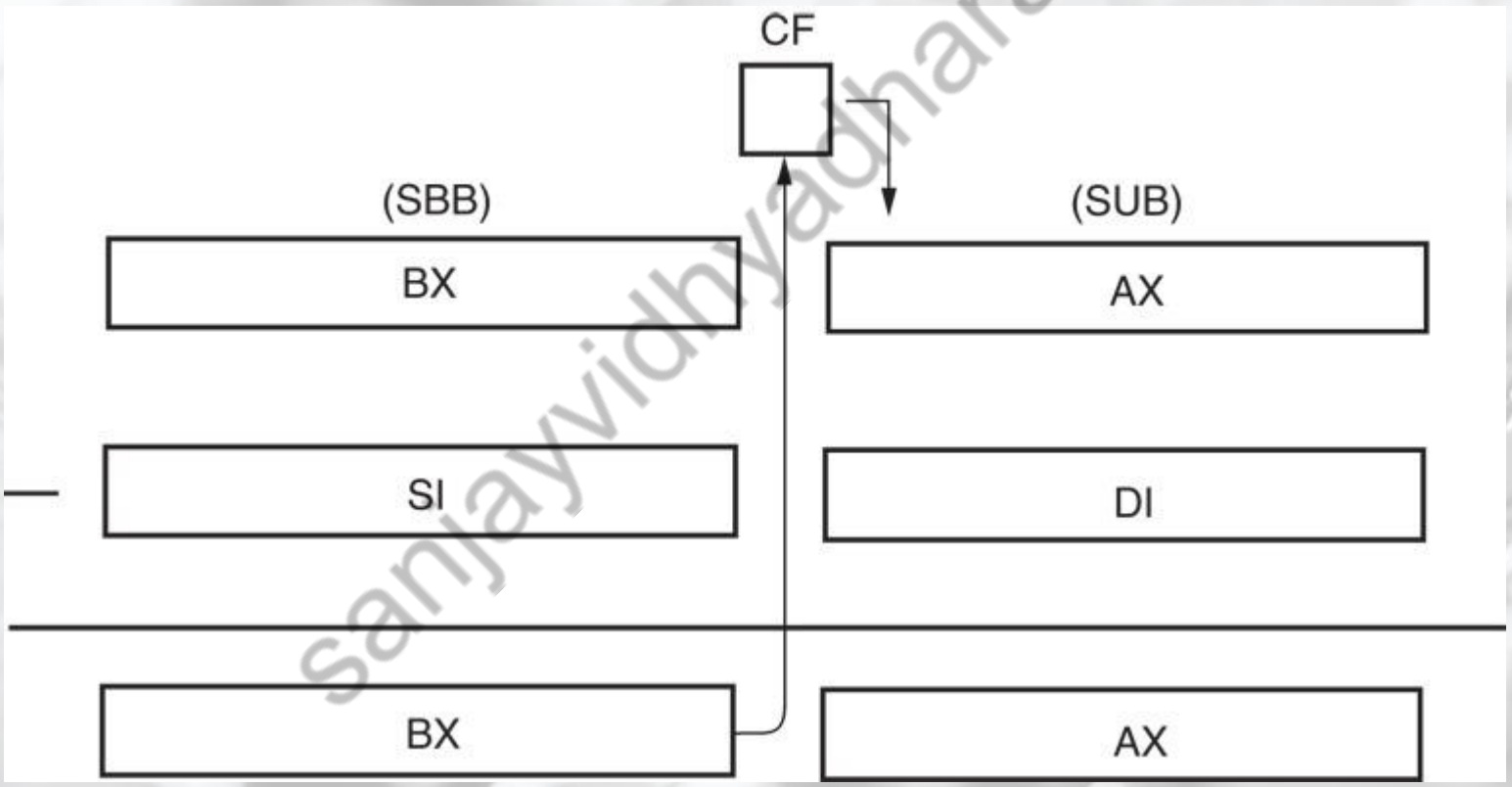
P = 1 (even parity),

AC = 0 (no half carry), O = 0 (no overflow).

sanjayvidhyadharan.in

Subtraction

Subtraction-with-borrow showing how the carry flag (C) propagates the borrow.



sanjayvidhyadharan.in

Subtraction

```
                                    2'compl.
 -1                  -1             --------> 1111 1111
44H 62H---> 0100 0100  0110 0010---------> 0100 0100  0110 0010
                                    2'compl.
-13H F1H--->-0001 0011 -1111 0001-------->+1110 1101 +0000 1111
 -------     ---------  ---------           ---------  ---------
 30H 71H---> 0011 0000  1111 0001           0011 0000  0111 0001(30H 71H)
```

C-Carry Flag This flag is set when there is a carry out of MSB in case of addition or a borrow in case of subtraction. For example, when two numbers are added, a carry may be generated out of the most significant bit position. The carry flag, in this case, will be set to '1'. In case, no carry is generated, it will be '0'. Some other instructions also affect or use this flag and will be discussed later in this text.

santavvidhyadharan.in

Increment

inc destination

Performs

destination := destination + 1

Increment instructions affect the flag bits, as do most other arithmetic and logic operations. The difference is that increment instructions do not affect the carry flag bit. Carry doesn't change because we often use increments in programs that depend upon the contents of the carry flag. Note that increment is used to point to the next memory element in a **byte-sized** array of data only

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

```
        clc
        mov     CX,0FFFFH
        add     CX,1
        jc      add_one
          .
          .
add_one:
        inc     DX
```

```
        clc
        mov     CX,0FFFFH
        inc     CX
        jz      add_one
          .
          .
add_one:
        inc     DX
```

sanjayvidhyadharan.in

Decrement

dec destination

Performs

destination := destination - 1

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

Negate

neg destination

Performs

destination := 2's complement of destination

sanjayvidhyadharan.in

Increment/Decrement

Example 1:

Copy the contents of a block of memory (100 bytes) starting at location 10100h to another block of memory starting at 10200h

```
        MOV AX,1000h
        MOV DS,AX
        MOV SI, 100
        MOV DI, 200
        MOV CX, 64h
NEXT:   MOV AL, [SI]
        MOV [DI], AL
        INC SI
        INC DI
        DEC CX
        JNZ NEXT
```

sanjayvidhyadharan.in

PTR

Is this byte, word or double word ?
INC [50h] ;
MOV [SI], 10h

To remove ambiguity, we use the PTR operator

INC BYTE PTR [50h]
INC WORD PTR [50h]
INC DWORD PTR [50h]
MOV byte ptr [SI], 10h
MOV word ptr[SI],10h

sanjayvidhyadharan.in

Compare Instruction

CMP Destination, Source

Performs

destination - source

Compare instruction is a subtraction that changes only the flag bits and operand but does not affect any of the operands

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

Ex: CMP AL, DL

CMP DX, 2000H

CMP [SI], CH

A CMP is normally followed by a conditional jump instruction, which tests the condition of the flag bits

sanjayvidhyadharan.in

Thank You

sanjayvidhyadharan.in