University of Salahaddin – College of Engineering
Software & Informatics Dep.

Computer Architecture II
2023-2024
Lecture 4

Lecturer Nyan D. Sallman

OUTLINE

- **Statement in assembly language**
- **Arithmetic Instruction**
- **Special Assembler Directive**
- **Basic logical Instructions**

Statement in assembly language

- Any statement consist of four part:

Label OP code Operand(s) Comment

Label :- symbolic name for memory location (begin with any letter or symbol :- a, b, A, B, …. @, $, -, ?,…….)

Op code:- code of the instruction

Operand:- data used by the instruction

Comment :- contains comments on instr., Previously begin with (;)

Variable definition in data segment

```
DATA1   DB      23H          ;label DATA1  defined as byte  contain 23h

DATA2   DW 1000H         ; label DATA2 defined as word  contain 1000h

Dir    DD   F100 F342h    ; label Dir defined as double word  contain
                          ; F100 F342h  (hold 4 byte in memory)
```

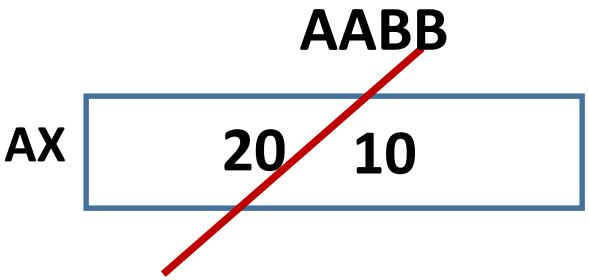
Example

```
Dt1     DB     23H       ;data1 defined as byte  contain 23h
Dt2     DW  1000H        ;data2 defined as word  contain 1000h

START:  MOV   AL,44H      ;copy 44 to AL
        MOV   CX, 200     ;copy 200 to CX

        MOV  Dt1, AL   ;copy AL into byte memory
                                location Dt1
        MOV BX, Dt2    ; copy word begin from memory
                                location named Dt2 to BX
        MOV ES:[2000h], BH
```

Example

```
        .DATA
DATA1   DB    10H
DATA2   DB    20H
DATA3   DW    1000h
DATA4   DW    AABBH
      .CODE
      .STARTUP
        MOV   AL,DATA1
        MOV   AH,DATA2
        MOV   DATA3,AX
        MOV   AX,DATA4
      .EXIT
```

2010



- **Some of illegal instruction :**

```
MOV  ES, DS                    ; segment to segment
MOV BL,DX                      ;mixed sizes
MOV CS,AX          ;CS must not to be destination register


MOV DATA1, DATA2               ;memory to memory directly
MOV ES:[DI],DS:[SI]            ;      = = =
```

Arithmetic Instruction

- Addition
- Subtraction
- Multiplication and Division

Basic logic instruction

- AND
- OR
- Exclusive-OR
- NOT
- SHIFT and ROTATE

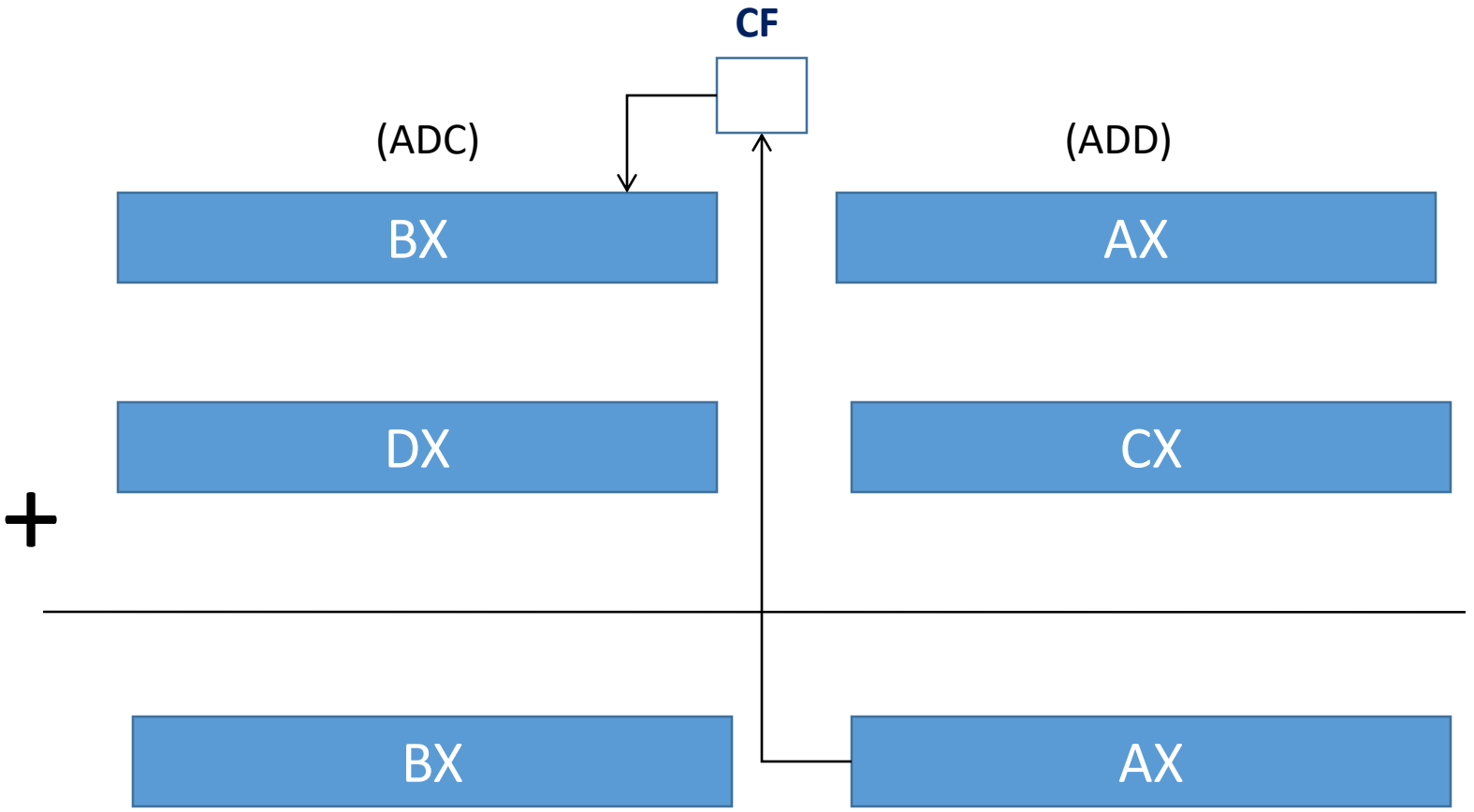
Addition operation

```
ADD AL,BL          ; AL=AL+BL
ADD CX,DI          ;CX=CX+DI
ADD CL,[BP]        ;CL=CL+[BP]
ADD  BH,33H        ;BH=BH +33H

ADC  BX,CX         ;BX=BX + CX+ carry
ADC  DX,[BP+2]     ;DX=DX+[BP+2]+ carry
                          stack
INC  BL            ; BL= BL+1
INC  SP            ; SP=SP+1
INC  DATA          ; DATA=DATA+1
```

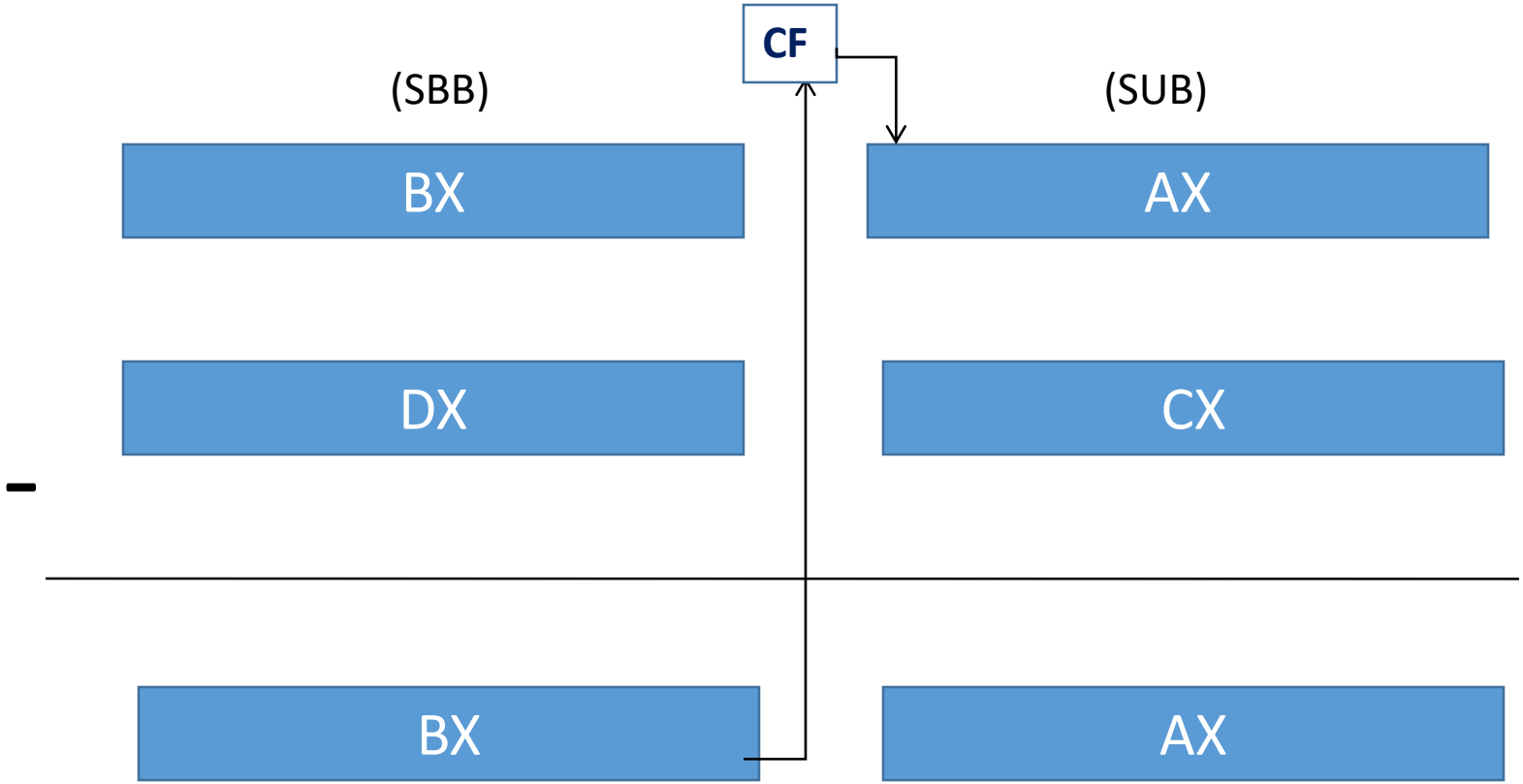
Subtraction operation

```
SUB  CL,BH          ; CL=CL-BH
SUB  AX,SP          ; AX=AX-SP
SUB  [BX],DX        ; [BX]=[BX]-DX
SUB  AH,TEMP        ;AH=AH- TEMP


SBB  AH,AL          ; AH=AH-AL-carry
SBB  DX,2           ;DX=DX-2-carry

DEC  BL             ;BL=BL-1
DEC   CX            ;CX=CX-1
DEC  NUMB1          ;NUMB1=NUMB1-1
```

Write a program to adds BX-AX with DX-CX and the sum appearing in BX-AX



Write a program to subtract DX-CX from BX-AX with the result appearing in BX-AX



Subtraction of 32 bit from 32 bit in 8086 microprocessor

```
MOV BX, AA00H
MOV  AX, 1122H
MOV  DX,1000H
MOV CX,2442H
SUB AX,CX
SBB BX,DX
```

```
    AA00   1122
-   1000   2442
```

Final result of (BX AX)-(DX CX)=(99FF ECE0)H

And the state of FLAGES : Z=0, S=1, AC=0, C=0, P=1 , and O=0

Special Assembler Directive
(PTR operator , OFFSET directive)

The PTR Operator

- INC [20h] ; is this byte/word/dword? or
- MOV [SI],5
  - Is this byte 05?
  - Is this word 0005?
  - Or is it double word 00000005?
- Byte or word or doubleword?

Improper instr. without operand size boundaries

- To clarify we use the PTR operator
  - INC BYTE PTR [20h]
  - INC WORD PTR [20h]
  - INC DWORD PTR [20h]
- or for the mov example:
  - MOV byte ptr [SI],5
  - MOV word ptr[SI],5

Solution by PTR operator

Q: Check on these instructions if there would use Pointer operator {PTR} or not

```
.data
LAB1    DB    10h
LAB2   DW    1000 H
.code
MOV    AL,  LAB1
MOV     DL, [BX]
SUB      [BX],2
MOV    CL,LAB2
ADD     AL, LAB1+1
```

```
.data
LAB1    DB    10h
LAB2   DW    1000 H
.code
MOV    AL, LAB1
MOV     DL, [BX]
SUB     BYTE PTR [BX], 2
MOV    CL, BYTE PTR LAB2
ADD     AL, LAB1 +1
```

OFFSET directive

```
0000  DATA   DW   1234H

      MOV BX, DATA              ;copy content of memory
                                 location DATA to BX
                                          BX=1234h

      MOV BX, OFFSET DATA  ;copy the offset address of
                                 DATA to BX
                                          BX=0000h
```

Multiplication

8- bit multiplication

- MUL CL ; AX=AL * CL (unsigned result)
- IMUL DH ; AX=AL*DH (signed result)
- IMUL BYTE PTR[BX] ; AX=AL*BYTE PTR[BX] (signed result)
- MUL TEMP ; AX=AL*TEMP (unsigned result)

Note: multiplicand always in AL

C, O flag are predictable and used

Carry	Overflow	Result
0	0	Most significant byte (8 bit multiplication), or most significant word (16 bit multiplication), is zeros
1	0	16 bit wide (8 bit multiplication), 32 bit wide (16 bit multiplication),

Example: multiply content of BL=5, and CL =10 and save the product in DX, suppose all the data unsigned:

Sol//

```
MOV BL,5
MOV CL,10
MOV AL,CL
MUL BL
MOV DX,AX
```

16- bit multiplication

- MUL CX ; DX_AX= AX *CX unsigned product
- IMUL DI ;DX_AX=AX *DI signed product
- MUL WORD PTR [SI]

;DX_AX=AX*WORD PTR [SI] unsigned product

Note : AX contain multiplicand

32- bit multiplication

- MUL ECX
- IMUL EDI
- MUL DWORD PTR[ESI]

;same operation as above, but multiplicand in EAX and the product to be save in EDX-EAX

(8 bit Division)

Dividend ⟶ AX

Divisor ⟶ 8-bit reg. or memory location

Result : AL= quotient

AH= Remainder

(8 bit Division)

- DIV CL
- IDIV BL
- DIV BYTE PTR [BP]

Example : if AX=0010h(+16), BL=FDh(-3)
After IDIV BL making AX=01FBh
Quotient AL= FB(-5) ; truth sign
Remainder AH =1 ; dividend sign
Note if the dividend (-16) and divisor(+3)
AL=(-5) and AH=(-1)

(16 bit Division)

Dividend ⟶ DX_AX

Divisor ⟶ 16-bit reg. or memory location

Result : AX= quotient

DX= Remainder

(16 bit Division)

- DIV CX ; (DX_AX) / CX , DX=Rem. , AX=Quo.
- IDIV SI
- DIV NUMB

Note: some Division need to extend AX

By:

CBW :- convert byte to word

CWD:- convert word to double

Example : Write Ass. Por. to Divide (AX=-100) on (CX=9)

```
MOV AX,-100
MOV CX , 9
CWD  ; convert word in (AX) to double word saved in DX-AX
IDIV CX
```

(32 bit Division)

Dividend ⟶ EDX-EAX

Divisor ⟶ 32-bit reg. or memory location

Result : EAX= quotient

EDX= Remainder

- DIV ECX
- IDIV DATA
- DIV DWORD PTR[EDI]

- Example : (program to save quotient and Remainder as a fraction value)

```
MOV AX,13
MOV BL,2
DIV BL
MOV ANSQ, AL    ; SAVE QUOTIENT
MOV AL,0
DIV BL          ;GENERATE REMAINDER
MOV ANSF,AL     ;SAVE Fraction
```