WEEK 3 LESSON 2
CHAPTER 4 PART-2

BASIC ARITHMETIC INSTRUCTIONS

PREPARED BY

AHMED AL MAROUF

LECTURER

DEPT. OF CSE

DAFFODIL INTERNATIONAL UNIVERSITY

OUTLINE

- Few Basic Instructions (Revising)
- Arithmetic Instructions
- Translation of High-level Language to Assembly Language
- Related Exercises

FEW BASIC INSTRUCTIONS

MOV instruction:

- The MOV instruction is used to transfer data between registers, between a register and a memory location, or to move a number directly into a register or memory location.

Syntax: **MOV destination, source**

Example:

```
MOV AX, WORD1      ; moving the value/content of word1 into AX
MOV AX, BX         ; moving the content of BX register into AX
MOV AX, 'A'        ; moving the ASCII value (41h) of character 'A'
```


SWAPPING/EXCHANGING CONTENTS OF TWO REGISTERS

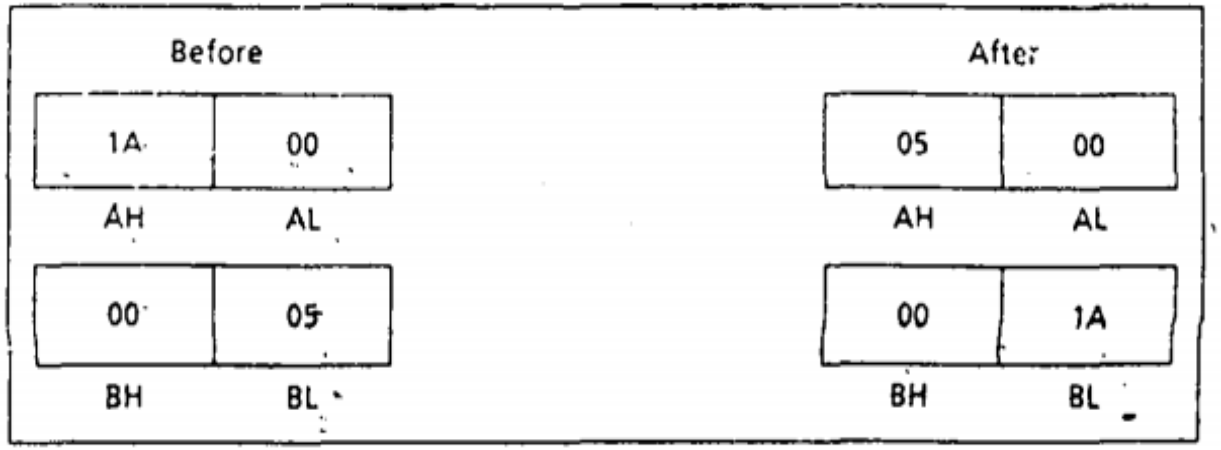
- Lets say AX contains 1234h and BX contains 5678h. How we have to swap/exchange the values of AX and BX.
- Possible way to do this: Using another register (CX or DX) for temporary use

```
MOV CX, AX
MOV AX, BX
MOV BX, CX
```

Or lets see how XCHG works….

XCHG INSTRUCTION

- Syntax: XCHG destination, source
- Example: XCHG AH, BL



- To do the same task of the last swapping example,

we can simply write,

XCHG AX, BX

ARITHMETIC INSTRUCTIONS

- Basic arithmetic operations can be performed in 8086.
- For Addition, Subtraction, Increment, Decrement, Negation, in 8086 we have separate instructions.
- "ADD" instruction is used for adding two values/contents of registers.

Syntax:

ADD destination, source

- Content of destination is added with the content of source and the result is stored in destination operand.

Example: ADD WORD1, AX

Content of WORD1 is added with the content of AX and the result is stored in WORD1.

ADD WORD1, AX

ADDITION INSTRUCTION

Legal and illegal ADD instructions:

- **Case 1:**

```
ILLEGAL: ADD  BYTE1,BYTE2
```

A solution is to move BYTE2 to a register before adding, thus

```
MOV AL,BYTE2          ;AX gets BYTE2
ADD BYTE1,AL          ;add it to BYTE1
```

- **Case 2:**

ADD 5, AX ;Illegal instruction

Reason: As the destination operand is not a register or memory location. It is an immediate value, which cannot store contents.

SUBTRACTION INSTRUCTION

SUB destination, source

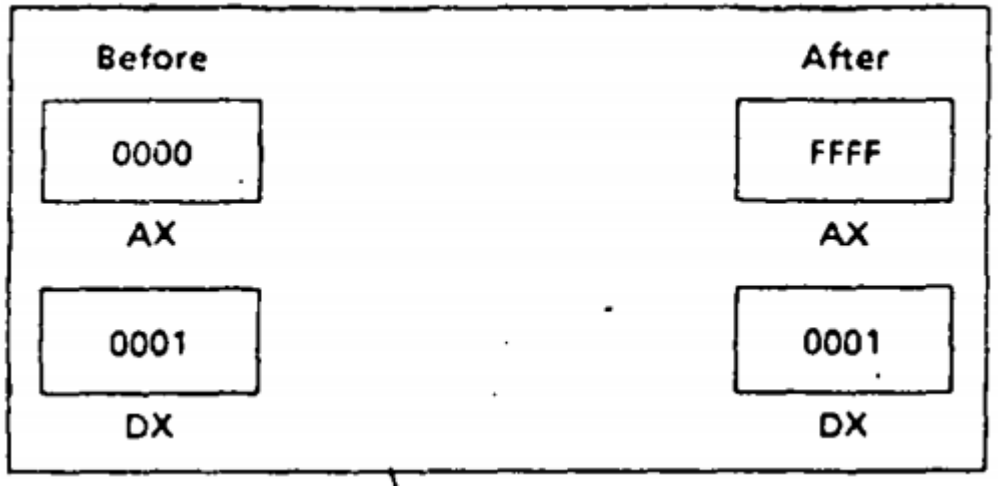
- The content of source operand is subtracted from the content of destination operand and the result is stored in the destination operand.

Example:

- **SUB AX, DX**

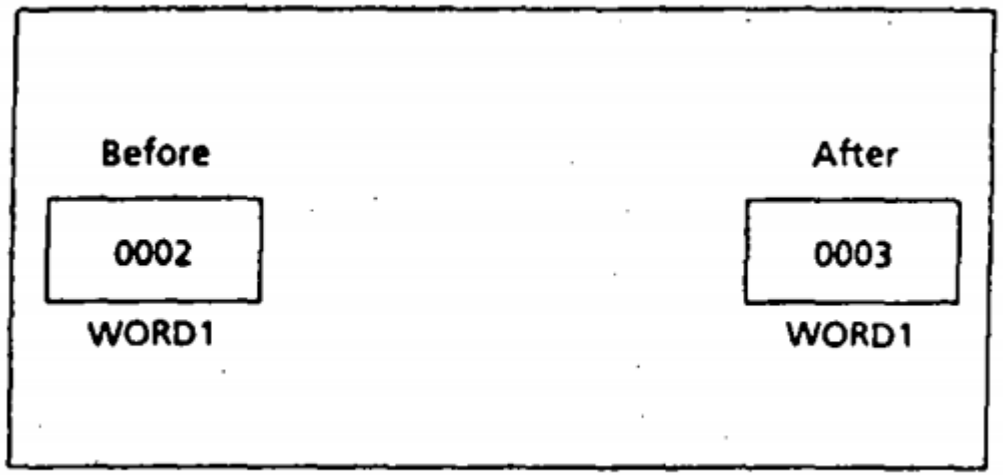
This instruction will do AX-DX and the result will be stored in AX register.

SUB AX, DX



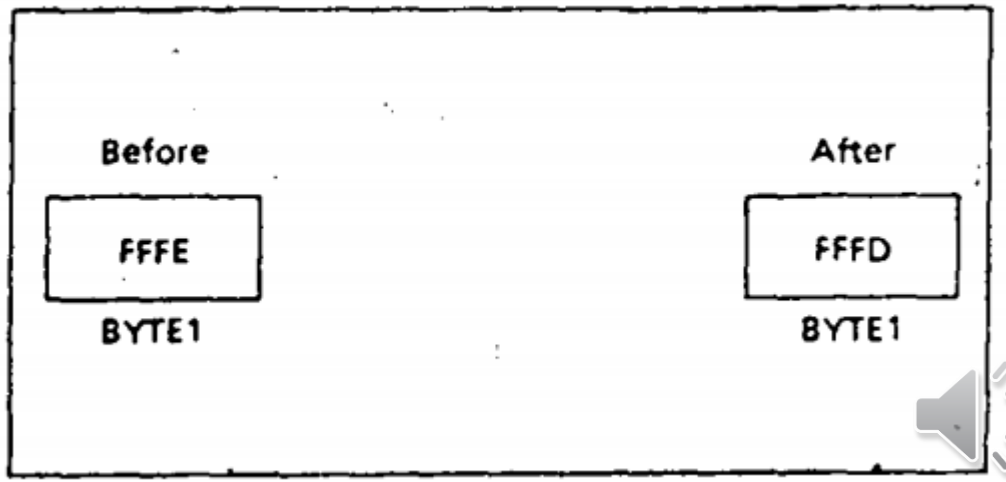
INCREMENT & DECREMENT INSTRUCTIONS

INC Word1



Just like i++ and i-- operations in C programming

DEC Byte1



NEGATION INSTRUCTION

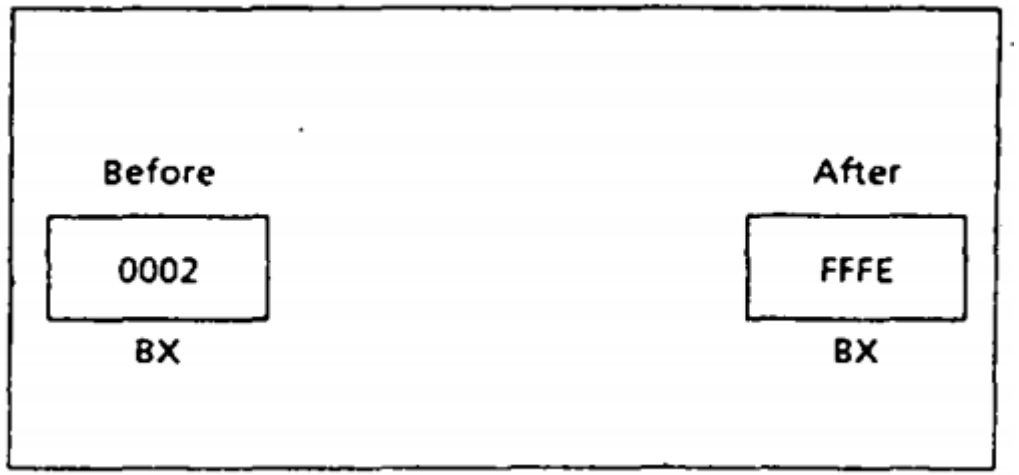
- "***NEG***" is used to negate the contents of the destination,
- NEG does this by replacing the contents by its two's complement.

Syntax: NEG destination

- The destination may be a register or memory

Example: NEG BX

- This instruction will negate the content of BX.



TRANSLATION OF HIGH-LEVEL LANGUAGE TO ASSEMBLY LANGUAGE

- Statements given in high-level language (i.e. C, C++, JAVA) are to be converted into assembly language instruction.
- Using only MOV, XCHG, ADD, SUB, INC, DEC, NEG instructions.

Example 1:

Statement	*Translation*	
B = A	MOV AX,A	;move A into AX
	MOV B,AX	;and then into B

Example 2:

A = 5 – A	MOV AX,5	;put 5 in AX
	SUB AX,A	;AX contains 5 - A
	MOV A,AX	;put it in A

Alternate Solution:

```
NEG A.          ;A = -A
ADD A,5         ;A = 5 - A
```

DO EXERCISE

- Chapter 4 Exercise 6

6. Using only MOV, ADD, SUB, INC, DEC, and NEG, translate the following high-level language assignment statements into assembly language. A, B, and C are word variables.

   a. A = B – A

   b. A = –(A + 1)

   c. C = A + B

   d. B = 3 × B + 7

   e. A = B – A – 1

THANK YOU