WEEK 3 LESSON 1
CHAPTER 4 PART-1
INTRODUCTION TO ASSEMBLY LANGUAGE

PREPARED BY

AHMED AL MAROUF

LECTURER

DEPT. OF CSE

DAFFODIL INTERNATIONAL UNIVERSITY

OUTLINE

- Assembly Instructions Syntax
  - Name Field
  - Operation Field
  - Operands Field
  - Comment Field
- Program Data
- Variables
  - Byte, Word and Array Variables
- Named Constants
- Few Basic Instructions
  - MOV and XCHG

ASSEMBLY INSTRUCTION SYNTAX

- Assembly language programs are translated into machine language instructions by an assembler.
- They must be written to confirm to the assembler's specifications (syntax).
- Assembly language code is generally not case sensitive
  - but we use upper case to differentiate code from the rest of the text.
- Here, we are going to use the ***Microsoft Macro Assembler (MASM).***

ASSEMBLY STATEMENTS

- Programs consist of ***statements***, one per line.
- Each statement is either an instruction or an ***assembler directive***.
- Assembler directives instruct the assembler to perform some specific task, such as allocating memory space for a variable or creating a procedure.

- Generic fields for each instructions:

Name	**Operation**	**Operand (s)**	**Comment**
Optional	**Mandatory**	**Depends on the operation**	**Optional**

NAME FIELD

- The name field is used for instruction labels, procedure names and variable names.
- The assembler translates names into memory addresses.

Rules:

- Can be from 1 to 31 characters long
- May consist of letters, digits and the special characters (?, @, _, %), except "&" sign
- Embedded blacks are not allowed
- If a period (full stop) is used, it must be the first character.
- May not begin with a digit.
- The assembler does not differentiate between upper and lower case in a name.

***Legal Names*:**

- COUNTER1
- ?character
- SUM_OF_DIGITS
- $1000
- DONE?
- .TEST

Illegal Name:

- TWO WORDS (contains a blank)
- 2abc (begins with a digit)
- A45.28 (. Not the first character)
- YOU&ME (contains an illegal character)

OPERATION FIELD

- For ***an instruction***, the operation field contains a symbolic ***operation code*** (***opcode***).
- The assembler translates a symbolic opcode into a machine language opcode.
- Opcode symbols often describe the operations function
  - For Example, MOV, ADD, SUB

- For ***an assembler directive***, the operation field contains a ***pseudo-operation code*** (***pseudo-op***)
- Pseudo-ops are not translated into machine code.
- They simple tell the assembler to do something.
  - For example, PROC pseudo-op is used to create a procedure.

OPERANDS FIELD

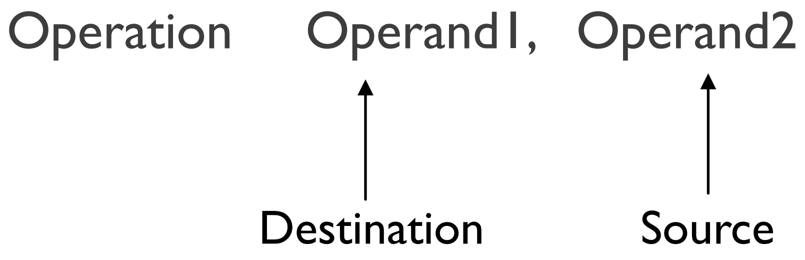
- For an instruction, the operand field specifies the data that are to be acted on by the operation.
- An instruction may have zero, one, or two operands.

For Example,

NOP ; no operands; does nothing

INC AX ; one operand; adds 1 to the contents of AX

ADD AX, BX ; two operand, adds the contents of AX and BX, stores in AX



COMMENT FIELD

- It is an optional field.
- Starts with a semicolon (;)
- Works only for one line comments.
- No multiple line comments available for 8086. We have to use semicolon for each line.

Example:

```
MOV CX, 0   ; move 0 to CX
```

It is a comment

PROGRAM DATA

- The processor operates only on binary data.
- The assembler translate all data representation into binary numbers.
- However, in an assembly language program ***we may express data as binary, decimal, or hex numbers, and even as characters***.
- Let us know about the following program data types: ***Numbers and Characters***
- Characters and strings must be enclosed in single or double quotes.
- For example: "A" or 'Hello'
- Characters are translated into ASCII codes by the assembler, So, "A" is same as 41h in a program.

Notations for Numbers:

- Numbers ending with "B" or "b" are in binary
- Numbers ending with "D" or "d" are in decimal
- Numbers ending with "H" or "h" are in hexadecimal
- Octal numbers are not allowed in 8086.

LETS FIND THE TYPE AND VALIDITY OF THE FOLLOWING NUMBERS

Numbers	Type
11011	Decimal
11011B	Binary
64223	Decimal
-21843D	Decimal
1,234	Illegal-contains a non-digit character
1B4DH	Hexadecimal
1B4D	Illegal hex number- doesn't end with "H" or "h"
FFFFH	Illegal hex number- doesn't begin with a decimal digit
0FFFFH	Hexadecimal

VARIABLES AND NAMED CONSTANTS

- DB (Define Byte), DW (Define Word), DD (Define Doubleword), DQ (Define quadword) and DT (Define tenbytes) are the pseudo-ops for variables.
- EQU (equates) is used for named constants.

- To be covered in LAB in details.

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

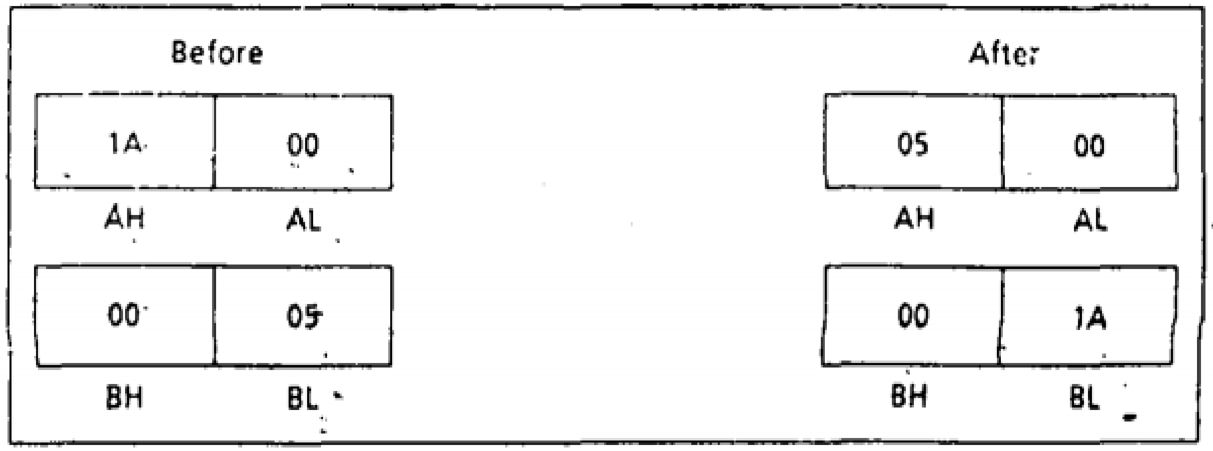
```
MOV CX, AX
MOV AX, BX
MOV BX, CX
```

Or lets see how XCHG works….

XCHG INSTRUCTION

- Syntax: XCHG destination, source

- Example: XCHG AH, BL

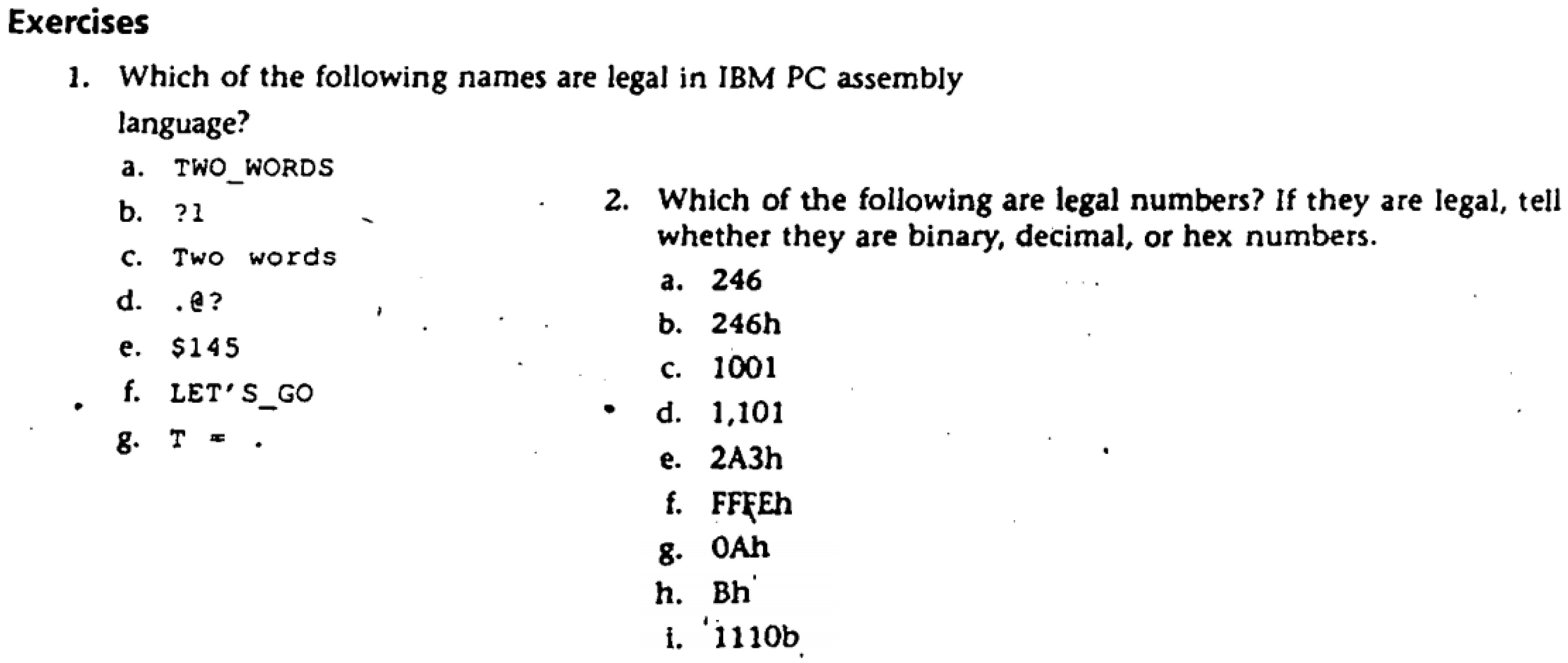


- To do the same task of the last swapping example,

we can simply write,

XCHG AX, BX

DO EXERCISE



Exercises

1. Which of the following names are legal in IBM PC assembly language?
   a. `TWO_WORDS`
   b. `?1`
   c. `Two words`
   d. `.@?`
   e. `$145`
   f. `LET'S_GO`
   g. `T = .`

2. Which of the following are legal numbers? If they are legal, tell whether they are binary, decimal, or hex numbers.
   a. 246
   b. 246h
   c. 1001
   d. 1,101
   e. 2A3h
   f. FFFEh
   g. 0Ah
   h. Bh
   i. 1110b

THANK YOU