CSE 331L / EEE 332L: Microprocessor Interfacing & Embedded System

Section: 7&9, Spring 2020

Lab - 02 (Input-Output Functions)

Variables:

- Declaration:

  Variable _Name DB/DW initial_value

  - DB = Define Byte
    Example:
    VAR1 DB 5EH

  - DW = Define Word
    Example:
    VAR2 DW 8DC6h

  - DB/DW are variable types

- Variable _Name:

  Can be any letter or digit combination, though it **must start with a letter.**
  It's possible to declare unnamed variables by not specifying the name using "?" (this variable will have an address but no name)

- Initial_value:

  can be any numeric value:

  - Decimal number ends with an optional "D"/"d"
  - Binary number ends with "B"/"b"
  - Hex number ends with **"H"/"h" and must start with a decimal digit.** Otherwise assembler would be unable to decide whether the data represents a number or a string. (Ex: 2AH, 5C4H, 1ABCH, 0ABCDH)
  - Numbers may have signs
  - "?" denotes an uninitialized byte/word

Example: Define a byte variable with the value 35H and print the value of the variable.

```
ORG 100H

.MODEL SMALL
.STACK 100H

.DATA
        VAR DB 35H

.CODE
MAIN PROC
        MOV AX, @DATA
        MOV DS, AX

        MOV AH, 2
        MOV DL, VAR
        INT 21H

        MOV AH, 4CH
        INT 21H
MAIN ENDP

END MAIN
```

@DATA is the name of the data segment defined by .DATA. The assembler translates the name @DATA into a segment number. Two instructions are needed because a number (the data segment number) may not be moved directly into a segment register.

Creating Constants:

Constants are just like variables, but they exist only until your program is compiled (assembled) because no memory is allocated for EQA names. After definition of a constant its value cannot be changed. To define constants EQU (equates) directive is used:

constant_name EQU constant_value (numeric value / string)

Example:

k EQU 5

MOV BX, k

Creating Arrays:

- Arrays can be seen as chains of variables. A text string is an example of a byte array, each character is presented as an ASCII code value (0-255).
  Example:

  a DB 48h, 65h, 6Ch, 6Ch, 6Fh, 00h
  b DB 'Hello', 0

- You can access the value of any element in an array using square brackets
  Example:

  MOV AL, a[3]

- You can also use any of the memory index registers BX, SI, DI, BP
  Example:

  MOV SI, 3
  MOV CL, a[SI]

- If you need to declare a large array you can use DUP operator.
  The syntax for DUP:
  NUMBER DUP (VALUE(S))
  number - number of duplicates to make (any constant value).
  value - expression that DUP will duplicate.

  Example:

  c DB 4 DUP(9) ;is an alternative way of declaring: c DB 9, 9, 9, 9

  d DB 4 DUP(1, 2) ;is an alternative way of declaring: d DB 1, 2, 1, 2, 1, 2, 1, 2

Example: Define a byte variable with a string “hello” and print it.

```
ORG 100H

.MODEL SMALL
.STACK 100H


.DATA

        VAR DB 5H
        MSG1 DB "HELLO!$"

.CODE

MAIN PROC
        MOV AX, @DATA
        MOV DS, AX

        MOV AH, 2
        MOV DL, VAR
        INT 21H

        MOV DL, 20H
        INT 21H

        MOV AH, 9
        LEA DX, MSG1
        INT 21H

        MOV AH, 4CH
        INT 21H
MAIN ENDP

END MAIN
```

INT 2lh, function 9, expects the offset address of the character string to be in DX. To get it there, we use a new instruction:

LEA Destination, Source

where destination is a general register and source is a memory location. LEA stands for "Load Effective Address." It puts a copy of the **source offset address** into the destination.

Example:

LEA DX, MSG1

puts the offset address of the variable MSG into DX.

Instruction	**Operands**	**Descriptions**
LEA	REG, MEM	Load Effective Address. Algorithm: REG = address of memory (offset)

Tasks

1. Create two arrays of size 5. Load one of the arrays with random numbers of your choice. The second array should be kept blank. Copy the contents of the first array into the second array in the same order. You must not use loops to accomplish this task.

2. Define a string with lower-case characters and print them in uppercase characters.