Laboratory Exercise #2
Addressing Modes

Target Course Outcome:
CO2: Designs a microprocessor-based firmware integrating a microprocessor with supporting peripherals and devices.

Objectives:
To use the different addressing modes supported by the 8086-instruction set in creating assembly language programs.

To understand the use of data transfer instructions in various addressing modes

Tools Required:
Emu8086 Emulator

Part 1:
Most assembly language instructions require operands to be processed. An operand address provides the location, where the data to be processed is stored. Some instructions do not require an operand, whereas some other instructions may require one, two, or three operands.

When an instruction requires two operands, the first operand is generally the destination, which contains data in a register or memory location and the second operand is the source. Source contains either the data to be delivered (immediate addressing) or the address (in register or memory) of the data. Generally, the source data remains unaltered after the operation.
The different addressing modes are as follow

- Immediate addressing
- Register addressing
- Direct addressing
- Register indirect addressing
- Base addressing
- Indexed addressing
- Based index addressing
- Based index with displacement addressing

Immediate Addressing
An immediate operand has a constant value or an expression. When an instruction with two operands uses immediate addressing, the first operand may be a register or memory location, and the second operand is an immediate constant. The first operand defines the length of the data.

Example:

```
MOV AX, 0123H; the immediate value 0123 (word) is transferred to the destination register AX
MOV AL, 05H    ; the immediate value 05 (byte) is transferred to the destination register AL (LB)
```

The first instruction shows the 16-bit value is transferred to register AX, while the 2nd instruction shows the 8-bit value is transferred to the lower byte of Register AX (that is AL)

Register addressing mode
It means that the register is the source of an operand for an instruction.

Example

```
MOV CX, AX  ;  copies the contents of the 16-bit AX register into
ADD BX, AX  ; adds the contents of the register AX to BX and stores the result in register BX
```

Direct addressing mode
The addressing mode in which the effective address of the memory location is written directly in the instruction.
Example

```
MOV AX, [1592H]
MOV AL, [0300H]
```

In the first instruction, the contents of memory location 1592H (2 bytes) is transferred to Register AX.
In the second instruction, the contents of memory location 0300H (1 byte) is transferred to the lower byte of Register AX (that is AL).

Register indirect addressing mode
This addressing mode allows data to be addressed at any memory location through an offset address held in any of the following registers: BP, BX, DI & SI. In this addressing mode the effective address is in SI, DI or BX.

Example

```
MOV AX, [BX]   ; Suppose the register BX contains 4895H, then the contents
               ; 4895H are moved to AX
```

Based addressing mode
In this addressing mode, the offset address of the operand is given by the sum of contents of the BX/BP registers and 8-bit/16-bit displacement.

Example

```
MOV DX, [BX+04]
ADD CL, [BX+08]
```

In this type of addressing mode, the effective address is the sum of base register and displacement.

Indexed addressing mode
In this addressing mode, the operand's offset address is found by adding the contents of SI or DI register and 8-bit/16-bit displacements.

Example
MOV BX, [SI+16]
ADD AL, [DI+16]

In this type of addressing mode, the effective address is sum of index register and displacement.

Based index addressing mode
In this addressing mode, the offset address of the operand is computed by summing the base register to the contents of an Index register.

Base register: BX, BP
Index register: SI, DI

Example
ADD CX, [AX+SI]
MOV AX, [AX+DI]

The physical memory address is calculated according to the base register.

Based indexed with displacement mode
In this addressing mode, the operands offset is computed by adding the base register contents. An Index registers contents and 8 or 16-bit displacement.

Example
MOV AX, [BX+DI+08]
ADD CX, [BX+SI+16]

In this type of addressing mode, the effective address is the sum of index register, base register and displacement.

Part II
Write a program to implement byte and word data transfer in different addressing modes.

Activity #1
Write a basic assembly language program using the **MOV** and **ADD** instructions and execute the code in the Emu8086 environment.

Procedure:
1. Start Emu8086 by selecting its icon from the start menu, or by running Emu8086.exe
2. Create a new file and label it as Exp2-1.1.asm
3. In the editor window, write the following instructions.

```
        ORG 100H

        MOV AL,25               ;       COPY 25H INTO 8 BIT AL REGISTER
        MOV AX,2345             ;       COPY 2345H INTO 16 BIT AX REGISTER
        MOV BX,AX               ;       COPY THE CONTENT OF AX INTO BX REGISTER(16 BIT)
        MOV CL,AL               ;       COPY THE CONTENT OF AL INTO CL REGISTER
        MOV AL,DATA1            ;       COPIES THE BYTE CONTENTS OF DATA SEGMENT MEMORY LOCATION
                                ;       DATA1 INTO 8 BIT AL
        MOV AX,DATA2            ;       COPIES THE WORD CONTENTS OF DATA SEGMENT MEMORY
                                ;       LOCATION DATA2 INTO 16 BIT AX
        MOV DATA3,AL            ;       COPIES THE AL CONTENT INTO THE BYTE CONTENTS OF DATA
                                ;       SEGMENT MEMORY LOCATION DATA3
        MOV DATA4,AX            ;       COPIES THE AX CONTENT INTO THE WORD CONTENTS OF DATA
                                ;       SEGMENT MEMORY LOCATION DATA4
        MOV BX,OFFSET DATA5     ;       THE 16 BIT OFFSET ADDRESS OF DS MEMEORY LOCATION DATA5 IS
                                ;       COPIED INTO BX
        MOV AX,[BX+0H]          ;       COPIES THE WORD CONTENT OF DATA SEGMENT MEMORY LOCATION
                                ;       ADDRESSED BY BX INTO AX(REGISTER INDIRECT ADDRESSING)
        MOV DI,02H              ;       ADDRESS ELEMENT
        MOV AX,[BX+DI]          ;       COPIES THE WORD CONTENT OF DATA SEGMENT MEMORY LOCATION
                                ;       ADDRESSED BY BX+DI INTO AX(BASE PLUS INDIRECT ADDRESSING)
        MOV AX,[BX+0002H]       ;       COPIES THE WORD CONTENT OF DATA SEGMENT (16 BIT)
        MOV AL,[DI+2]           ;       REGISTER RELATIVE ADDRESSING
        MOV AX,[BX+DI+0002H]    ;       COPIES THE WORD CONTENT OF DATA SEGMENT MEMORY LOCATION
                                ;       ADDRESSED BY BX+DI+0002H INTO AX(16 BIT)

        RET

DATA1   DB      25H
DATA2   DW      1234H
DATA3   DB      0H
DATA4   DW      0H
DATA5   DW      4321H,6789H
```

4. Press the Compile and Emulate button or press F5 key on your keyboard. The emulator window should open with this program loaded, click the Single Step button and watch the register values.

5. Write the value of the registers in Canvas after each instruction as seen on the emulator window.

Copyright Information

Author : Rosana J. Ferolin (rjferolin@usc.edu.ph)
Date of release : August 7, 2020
Version : 1.0

Change log:

Date	Version	Author	Changes
Aug. 7, 2020	1.0	Rosana J. Ferolin	Initial Draft
Sep. 2, 2025	2.0	Marlowe Edgar C. Burce	Corrected typo errors, modified code, procedures and activities