Laboratory Exercise #3-2
Logical Operations

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

Objectives:
To use the different arithmetic and logic operations in creating assembly language programs.
To study and implement the concept of arrays in Assembly Language Programming.

Tools Required:
Emu8086 Emulator

Part 1:

Logical Operations/Instructions

AND – AND Destination, Source
This instruction ANDs each bit in a source byte or word with the same numbered bit in a destination byte or word. The result is put in the specified destination. The content of the specified source is not changed.

The source can be an immediate number, the content of a register, or the content of a memory location. The destination can be a register or a memory location. The source and the destination cannot both be memory locations. CF and OF are both 0 after AND. PF, SF, and ZF are updated by the AND instruction. AF is undefined. PF has meaning only for an 8-bit operand.

Example:

```
AND BH, CL          ; AND byte in CL with byte in BH; Result in BH
AND BX, 00FFH       ; 00FFH Masks upper byte, leaves lower byte unchanged.
```

OR – OR Destination, Source
This instruction ORs each bit in a source byte or word with the same numbered bit in a destination byte or word. The result is put in the specified destination. The content of the specified source is not changed.

The source can be an immediate number, the content of a register, or the content of a memory location. The destination can be a register or a memory location. The source and destination cannot both be memory locations. CF and OF are both 0 after OR. PF, SF, and ZF are updated by the OR instruction. AF is undefined. PF has meaning only for an 8-bit operand.

Example:

```
OR AH, CL       ; CL ORed with AH, result in AH, CL not changed
```

```
        OR BP, SI       ; SI ORed with BP, result in BP, SI not changed
        OR SI, BP       ; BP ORed with SI, result in SI, BP not changed
        OR BL, 80H      ; BL ORed with immediate number 80H; sets MSB of BL to 1
```

XOR – XOR Destination, Source
This instruction Exclusive-ORs each bit in a source byte or word with the same numbered bit in a destination byte or word. The result is put in the specified destination. The content of the specified source is not changed.

The source can be an immediate number, the content of a register, or the content of a memory location. The destination can be a register or a memory location. The source and destination cannot both be memory locations. CF and OF are both 0 after XOR. PF, SF, and ZF are updated. PF has meaning only for an 8-bit operand. AF is undefined.

Example:

```
        XOR CL, BH      ; Byte in BH exclusive-ORed with byte in CL. Result in CL. BH not changed.
        XOR BP, DI      ; Word in DI exclusive-ORed with word in BP.  Result in BP. DI not changed.
```

NOT – NOT Destination
The NOT instruction inverts each bit (forms the 1's complement) of a byte or word in the specified destination. The destination can be a register or a memory location. This instruction does not affect any flag.

Example:

```
NOT BX          ; Complement content or BX register
```

NEG – NEG Destination
This instruction replaces the number in a destination with its 2's complement. The destination can be a register or a memory location. It gives the same result as the *invert each bit and add one* algorithm. The NEG instruction updates AF, AF, PF, ZF, and OF.

Example:

```
        NEG AL          ; Replace number in AL with its 2's complement
        NEG BX          ;Replace number in BX with its 2's complement
```

TEST – TEST Destination, Source
This instruction ANDs the byte / word in the specified source with the byte / word in the specified destination. Flags are updated, but neither operand is changed. The test instruction is often used to set flags before a Conditional jump instruction.

The source can be an immediate number, the content of a register, or the content of a memory location. The destination can be a register or a memory location. The source and the destination cannot both be memory locations. CF and OF are both 0's after TEST. PF, SF and ZF will be updated to show the results of the destination. AF is undefined.

Example:

```
        TEST AL, BH             ; AND BH with AL. No result stored; Update PF, SF, ZF.
```

```
TEST CX, 0001H      ; AND CX with immediate number 0001H; No result stored; Update PF,
                      SF, ZF
TEST BP, [BX][DI]   ; AND word is offset [BX][DI] in DS with word in BP.  No result stored.
                      Update PF, SF, and ZF
```

Summary of logical operations

OPCODE	OPERAND	DESTINATION	EXAMPLE
AND	D, S	D = D AND S	AND AX, 0010
OR	D, S	D = D OR S	OR AX, BX
NOT	D	D = NOT of D	NOT AL
XOR	D, S	D = D XOR S	XOR AL, BL
TEST	D, S	performs bit-wise AND operation and affects the flag registor	TEST [0250], 06
SHR	D, C	shifts each bit in D to the right C times and 0 is stored at MSB position	SHR AL, 04
SHL	D, C	shifts each bit in D to the left C times and 0 is stored at LSB position	SHL AX, BL
ROR	D, C	rotates all bits in D to the right C times	ROR BL, CL
ROL	R, C	rotates all bits in D to the left C times	ROL BX, 06
RCR	D, C	rotates all bits in D to the right along with carry flag C times	RCR BL, CL
RCL	R, C	rotates all bits in D to the left along with carry flag C times	RCL BX, 06

Part II

Write a program to implement the different logical operations.

Activity #1

Using registers and immediate value or immediate data .

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 Exp3-2.asm
3. In the editor window, write the following instructions.

```
ORG 100H

MOV AX,00H
MOV CL,05H
```

```
MOV SI,00H
BACK:
ADD AL, ARR[SI]
INC SI
DEC CL
JNZ BACK
MOV BL,05H
DIV BL

RET
ARR DB 01H,02H,03H,04H,05H
```

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 execution 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. 3, 2025	2.0	Marlowe Edgar C. Burce	Corrected arithmetic instruction syntax errors, revised code and procedures