

# EXPERIMENT 8

## Logic Instructions

### Objective

---

- Understand working of Logic instruction.

### Theory

---

#### Logic instruction

There are four logic instructions in 8086/88 assembly language. These instructions are AND (AND operation), OR (OR operation), XOR (Exclusive OR operation) and NOT (Inverting operation). The logic instruction is important to perform operation on some specific bit(s) and unchanged other bit(s) of the register.

#### The AND operation

The AND operation is used to clear bit or number of bits of the source register while all other bits remain unchanged. To clear bit write 0 and to make bit unchanged write 1 on the specific bit position.

E.g.

If upper nibble of the register is required and lower nibble is set to zero then AND operation is used as.

```
MOV  AL, 72 H    ; AL = 72 H
AND  AL, 0F0 H    ; AL = 70 H
```

Working

```
0111 0010 B  (72) H
AND  1111 0000 B  (0F0) H
```

```
-----
0111 0000 B  (70) H
-----
```

#### The OR operation

The OR operation is used to set bit or number of bits of source register while all other bits remain unchanged. To set bit write 1 and to make bit unchanged write 0 on the specific bit position.

E.g.

If an ASCII character (8-bit) is present in the register and wants to convert in lower case regardless the character is already in lower case.

```
MOV AL, 'A'      ; AL = 41H (ASCII of A)
OR  AL, 20H      ; AL = AL OR 20H
```

Working

If AL contain 41 H (0100 0001 B)

```
0100 0001 B (41) H
OR  0010 0000 B (20) H
```

```
-----
0110 0001 B (61) H      ASCII of 'a'
-----
```

And if AL contain 61 H (0110 0001 B)

```
0110 0001 B (61) H
OR  0010 0000 B (20) H
```

```
-----
0110 0001 B (61) H      ASCII of 'a'
-----
```

### The XOR operation

The XOR (Exclusive OR) operation is used to compliment (invert) specific bit or number of bits of the source register while all other bits remain unchanged. To invert bit write 1 and to make bit unchanged write 0 on the specific bit position.

E.g.

If upper nibble of the register is required to invert while lower nibble is want to remain unchanged then XOR operation is used as.

```
MOV AL, 0A5H      ; AL = A5 H
XOR AL, 0F0H      ; AL = 55 H
```

Working

```
1010 0101 B (A5) H
AND 1111 0000 B (F0) H
```

```
-----
0101 0101 B (55) H
-----
```

### The NOT operation

The NOR (Compliment or Inverting) operation is used to compliment (invert) the bits of register. In the NOT operation the 1 is change by 0 and 0 is change by 1.

E.g.

Compliment the bits in the register

```
MOV AL, 0AAH      ; AL = AA H
NOT AL             ; AL = 55 H
```

Working

NOT 1010 1010 B (AA) H

0101 0101 B (55) H

## Exercise

1. Write a program using Logical Gates which interact binary value.

org 100h

.model small

.data

.code

main proc

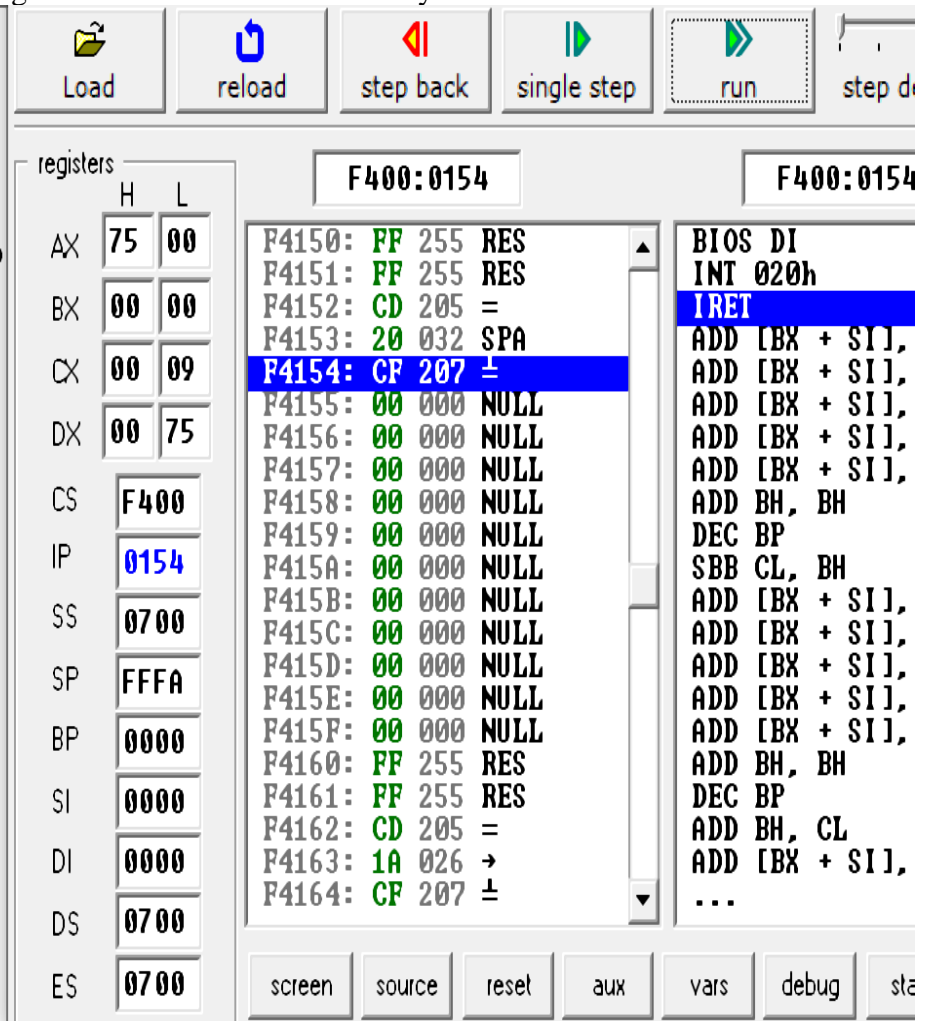
mov ah,10001010h

NOT ah

mov dl,ah

main endp

ret



The screenshot shows a debugger window with the following components:

- Assembly View:** Displays assembly instructions at memory addresses F400:0154 to F4164. The instruction at F4154 is highlighted: `F4154: CF 207 ±`.
- Registers View:** Shows the state of 16-bit registers. The AX register contains 75 00, BX contains 00 00, CX contains 00 09, and DX contains 00 75. Segment registers CS, SS, SP, BP, SI, DI, DS, and ES are also shown.
- Control Panel:** Includes buttons for Load, reload, step back, single step, run, and step over.
- Disassembly View:** Shows the disassembly of the selected instruction, including BIOS DI, INT 020h, and IRET.

2. Write a program that compute binary calculation in the form of binary calculator.

```
.model small
.data
.code
main proc

    mov ah,10001010b
    mov bh,11010111b

    AND ah,bh

    mov dl,ah

    sub dl,81
    mov ah,02
    int 21h

    add dl,2
    mov ah,02
    int 21h

    sub dl,3
    mov ah,02
    int 21h

    main endp

ret
```

Load reload step back single step run step c

registers

	H	L
AX	02	30
BX	D7	00
CX	00	20
DX	00	30
CS	F400	
IP	0154	
SS	0700	
SP	FFFA	
BP	0000	
SI	0000	
DI	0000	
DS	0700	
ES	0700	

F400:0154

F4150:	FF	255	RES
F4151:	FF	255	RES
F4152:	CD	205	=
F4153:	20	032	SPA
F4154:	CF	207	±
F4155:	00	000	NULL
F4156:	00	000	NULL
F4157:	00	000	NULL
F4158:	00	000	NULL
F4159:	00	000	NULL
F415A:	00	000	NULL
F415B:	00	000	NULL
F415C:	00	000	NULL
F415D:	00	000	NULL
F415E:	00	000	NULL
F415F:	00	000	NULL
F4160:	FF	255	RES
F4161:	FF	255	RES
F4162:	CD	205	=
F4163:	1A	026	→
F4164:	CF	207	±

F400:015

BIOS DI
INT 020h
IRET
ADD [BX + SI],
ADD [BX + SI],
ADD [BX + SI],
ADD [BX + SI],
ADD [BX + SI],
ADD BH, BH
DEC BP
SBB CL, BH
ADD [BX + SI],
ADD [BX + SI],
ADD [BX + SI],
ADD [BX + SI],
ADD [BX + SI],
ADD BH, BH
DEC BP
ADD BH, CL
ADD [BX + SI],
...

emulator screen (80x25 chars)

130