Experiment 6

6 Logical and Bitwise Instructions

Introduction

In this experiment, the shift rotate and logical instructions are introduced. Logical, shift and rotate instructions are known as bitwise operations. These operations are designed for low-level operations, and are commonly used for low-level control of input/output devices. A number of examples that use such instructions are discussed in this experiment.

Objectives

- Logical Instructions: AND, OR, XOR
- Shift Instructions: SHL, SHR,SAR, SAL
- Rotate Instructions: ROL, ROR, RCL, RCR
- Simulating Division and Multiplication
- Reading and Displaying binary numbers, Parity checking

6.1 Logical Instructions

The list of logical instructions is given in Table 7.1. Effects of these instructions on the flags are also shown.

Instruction	Example	Meaning	Flags					
			OF	**SF**	**ZF**	**AF**	**PF**	**CF**
AND	AND AX, FFDFH	AX ← AX **AND** FFDFH	0	*	*	?	*	0
OR	OR AL, 20H	AL ← AL **OR** 20H	0	*	*	?	*	0
XOR	XOR NUM1, FF00	[NUM1]←[NUM1] **XOR** FF00	0	*	*	?	*	0
NOT	NOT X	X ← **1's complement of** X	-	-	-	-	-	-

Table 7.1: Logic Instructions

Logical instructions are the software analogy of logic gates. They are commonly used to separate a bit or a group of bits in a register or in a memory location, for the purpose of testing, resetting or complementing a group of bits.

Example

If b is the value of the 4^{th} bit in the 8-bit memory variable X, which could also be a register, the effects of the basic operations on bit b are shown in Table 7.2. Assume the initial value of X isd A5h (1010 0101b), to access the 4^{th} bit f X the following masks are needed: 0000 1000b or 1111 0111b.

Instruction		Effect
AND X, 11110111b	**b AND 0 = 0**	Clear bit b
OR X, 00001000b	**b OR 1 = 1**	Set bit b to 1
XOR X, 00001000b	**b XOR 1 = b**	Complement bit b

Table 7.2: Effects of the basic logical instructions

To complement all bit of the variable X, the instruction NOT is used:

```
; X = A5h = 1010 0101b
NOT X
; X = 5Ah = 01011010b
```

6.2 The Shift Instructions

Shift instructions with their syntax are shown in table 7.3. When the number is to be shifted by 1 bit, this could be directly indicated in the instruction. However, if the number of shifts exceeds 1, the number of shifts is indicated in register CL.

Instruction	Example	Meaning	Flags					
			OF	SF	ZF	AF	PF	CF
SHL	SHL AX,1	Shift AX left by 1 bit. Fill vacated bit with 0.	*	*	*	?	*	*
SAL	SAL AX,1	Shift AX left by 1 bit. Fill vacated bit with 0.	*	*	*	?	*	*
SHR	SHR NUM2,CL	Shift NUM2 right by the number of bits indicated in CL. Fill vacated bits with 0.	*	0	*	?	*	*
SAR	SAR NUM2,CL	As SHR but fill vacated bits with the sign bit.	*	*	*	?	*	*

Table 7. 3: The Shift Instructions

Right shift using the SHR instruction of a signed number affects the sign bit, which could cause the number to change its sign. The Shift Arithmetic Right (SAR) instruction is used to manipulate signed numbers. SAR preserves the sign bit by filling the vacated bits with the sign of the number. Shift Arithmetic Left (SAL) is identical in operation to SAR.

Shift operations may be used to multiply or divide by powers of 2 (i.e. 2^n). Multiplication by 2 is achieved by a one-bit left shift, while division by 2 is achieved by a one-bit right shift.

Example

```
MOV CL, n
SHL AX, CL      ; equivalent to AX = AX * 2^n
SHR AX, CL      ; equivalent to AX = AX ÷ 2^n
```

6.3 The Rotate Instructions

Rotate Instructions are very similar to the shift operations, except that the bits are shifted out from one end of the number and fed back into the other end to fill the vacated bits. Parity of a number is therefore not affected. Rotate Instructions may be used to facilitate shifting of long numbers (i.e. numbers of more than 16 bits).

The rotate through carry instructions (RCL and RCR) are similar to the regular rotate instructions (ROL and ROR), but the carry flag is considered as part of the number. Hence, before the rotate operation, the carry flag bit is appended to the number as the least significant bit in the case of RCL, or as the most significant bit in case of RCR

Instruction	Example	Meaning	Flags					
			OF	SF	ZF	AF	PF	CF
ROL	ROL BH, CL	Same as SHL, but shifted bits are fed back to fill vacated bits.	*	-	-	-	-	*
RCL	RCL BH, CL	Same as ROL, but carry flag is appended as MSB, and its content is shifted with the number.	*	-	-	-	-	*
ROR	ROR NUM1, 1	Same as SHR, but shifted bits are fed back to fill vacated bits.	*	-	-	-	-	*
RCR	RCR NUM1, 1	Same as ROR, but carry flag is appended as LSB, and its content is shifted with the number.	*	-	-	-	-	

Table 7. 4: The Rotate Instructions

Example

Write a program that counts the number of ones in BX register. Make sure that the content of BX will remain unchanged.

```
        MOV BX, NUM
        MOV CX, 16
        XOR AL, AL          ; CLEAR AL
NXT:    ROL BX, 1           ; MSB IN THE CARRY FLAG
        ADC AL, 0           ; ADD CARRY TO AL
        LOOP NXT
        ; AL = number of ones in BX register
```

Similar programs that count the number of zeros, or test that parity of a variable can be written.

6.4 Number Manipulation

He example already discussed in experiment 5, are repeated here using shift and logical instructions. Such instructions are more useful when dealing with BCD numbers. Other bases are better dealt with using MUL and DIV instructions.

6.4.1 Byte manipulations for reading purposes

1 / To put two decimal digits in the same byte:

```
CALL READC
MOV BL, AL          ; Read high digit e.g. 8 and save it in BL
CALL READC          ; Read low digit e.g. 3, and leave it in AL
XOR AH, AH          ; Clear AH
MOV CL, 4           ; Use shift left by 4 bits
SHL BL, CL          ; Shift DH 4 bits to the left
OR BL, AL           ; Result in BL ← 83
; Read second number at this level and perform operation afterwards
DAA                 ; Assume result is in AL in BCD format
; Number in AL register. See next how to display it as decimal number.
```

The procedure READC is defined below:

```
READC PROC NEAR
        MOV AH, 01H
        INT 21H
        SUB AL, 30H
        RET
READC ENDP
```

6.4.2 Byte manipulations for displaying purposes

2 / To display a number in BCD format use the following:

```
; The number is in the AL register:
MOV DL, AL          ; Move AL to DL
MOV CL, 4
SHR AL, CL          ; Shift  AL 4 bits to the right
AND AL, 0FH         ; Clear high nibble of AL
ADD AL, 30H         ; Convert to character
MOV DL, AL
CALL DISPC          ; Now Display AL as the high digit first
MOV AL, DL          ; Get number again
AND AL, 0FH         ; Clear 4 high nibbles of AL
ADD AL, 30H         ; Convert to character
MOV DL, AL
CALL DISPC          ; Now Display AL as the low digit second
```

The procedure DISPC is defined below:

```
READC PROC NEAR
        MOV AH, 02H
        INT 21H
        SUB AL, 30H
        RET
READC ENDP
```

6.5 Lab Work

Pre Lab Work:

1. Study program 6.1, and explain how the program reads the 2 binary numbers, and how it displays the result in binary format.
2. Write, assemble, link and run program 6.1.
3. Bring your work to the lab.

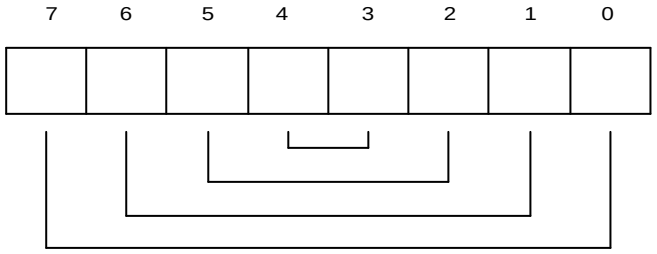
Lab Work:

1- Run program 6.1. Enter different binary numbers and see the displayed value on the screen.
2- Rewrite the program and make it read 16 bit numbers, adds them and displays the result in binary.
3- Write a program that prompts the user to enter two numbers of 4 digits each. Converts these each number to hexadecimal. Then calculates the sum and the difference of the two numbers, and finally displays the result in decimal format. Name it program 6.3. Use the program outlines given in program 6.2.
4- Show all your work to the instructor.
5- Submit all your work at the end of the lab session.

Lab Assignment:

1. Assume you have a 32 bit number in NH:NL, write a program that rotates the whole number by 1 bit to the left (or right). Display the result in binary on the screen, generate a delay using an empty for loop, rotate then display the number again on the same position.

2. Write a program that prompts the user to enter an 8 bit binary number, between 0 and 255. The program then inverts all bits according to the figure below. This program is useful in Signal Processing for the calculation of the Fast Fourier Transform (FFT). The operation is called Decimation (in time or frequency), and the bit manipulation is called bit shuffling i.e. rearrangement of bits.



Bit (i) ← Bit (7- i) for i = 0 – 7

Figure 6. 1: Bit Shuffling

TITLE "Program 6.1"

```
; This program adds 2 binary numbers and prints the result in binary format

.MODEL SMALL
.STACK 200
.DATA
        PROMPT1     DB      'Enter the first 8-bit binary number:  ','$'
        PROMPT2     DB      'Enter the second 8-bit binary number: ','$'
        PROMPT3     DB      'The sum of the two numbers in binary is: ','$'
        NUM1        DB      ?
        NUM2        DB      ?

.CODE
.STARTUP
        ; DISPLAY PROMPT1
        CALL READBIN8               ; READ AND CONVERT THE FIRST NUMBER
        MOV NUM1, BL        ; SAVE NUMBER1

        ; DISPLAY PROMPT2
         CALL READBIN8              ; READ AND CONVERT THE SECOND NUMBER
        MOV NUM2, BL                ; SAVE NUMBER 2
        MOV BL, NUM1
        ADD BL, NUM2                ; ADD THE TWO NUMBERS
        ; DISPLAY PROMPT3
        CALL DISPBIN8
.EXIT

;*************************************************
; READING A BINARY NUMBER
READBIN8    PROC    NEAR
            MOV BX, 0000
            MOV CX, 0008
            MOV AH, 01H
L1:         INT 21H
            SUB AL, 30H
            SHL BL, 1
            OR BL, AL
            LOOP L1
            RET
READBIN8  ENDP

;*************************************************
; DISPLAYING A BINARY NUMBER
DISPBIN8  PROC NEAR
            MOV CX, 0008                ; DISPLAYING
NEXT:       R0L BL, 1
            JNC BIT_0
            MOV DL, '1'
            MOV AH, 02H
            INT 21H
            JMP LAST
BIT_0:      MOV DL, '0'
            MOV AH, 02H
            INT 21H
LAST:       LOOP NEXT
            RET
DISPBIN8 ENDP
END
```

```
;************************************************
;************************************************
 TITLE "PROGRAM 6.2"
; This program shows how to manipulate a two-digit numbers

.MODEL SMALL
.STACK 200
.DATA
        NUM1 DB ?
        NUM2 DB ?

.CODE
.STARTUP

        ; READ NUMBER NUM1

        MOV AL, NUM1
        AND AL, 0FH
        MOV CL, 04H
        SHL AL,CL
        MOV BL, AL

        MOV AL, NUM1
        AND AL, 0F0H

        MOV CL, 04H
        SHR AL, CL

        OR BL, AL
        MOV NUM2, BL


        ; DISPLAY NUMBER NUM1

        ; CONVERT NUM2 TO BINARY
        ; DISPLAY NUMBER NUM2
.EXIT
END
```