Activity No. 6.1	
Bit Manipulation	
Course Code: CPE021	**Program:** Computer Engineering
Course Title: Computer Architecture and Organization	**Date Performed:** April 6, 2025
Section: CPE22S2	**Date Submitted:** April 6, 2025
Name: Adia, James Russel E.	**Instructor:** Engr. Maria Rizette H. Sayo

A. Procedure: Output(s) and Observation(s)

- **Objective**
  This activity aims to demonstrate bit manipulation in assembly language

- **Intended Learning Outcomes**
  After completion of this activity the students should be able to:
  1.1 Manipulate bits of data
  1.2 Compare the different bit manipulation instructions
  1.3 Create a program using the bit manipulation instructions

Sample Problem A.

1. Type the following program in a Notepad.

```
TITLE bit.asm
.model small
.stack 100h
.data
num db 03Dh
.code
main proc
movax,@data
movds,ax
movbl,num
mov cx,8
here: shr bl,1
Jcis_one
Mov dl,30h
Jmp print
ls_one:
Mov dl,31h
Print:
Mov ah,2
int 21h
loop here
Exit: Mov ax, 4c00h
Int 21h
```

Main endp

End main

```
TITLE bit.asm
.model small
.stack 100h

.data
num db 03Dh

.code
main proc
    mov ax, @data
    mov ds, ax
    mov bl, num
    mov cx, 8

here:
    shr bl, 1
    jc is_one
    mov dl, 30h
    jmp print

is_one:
    mov dl, 31h

print:
    mov ah, 2
    int 21h
    loop here

exit:
    mov ax, 4c00h
    int 21h

main endp
end main
```

1. Save the program as **bit.asm**.
2. Assemble and execute the program.

```
C:\HOA_6.1>tasm bit.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   bit.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  491k
```

```
C:\HOA_6.1>tlink bit.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

3. Analyze and record the output in Table 6.1.

Table 6.1 - Output of bit.asm

```
C:\HOA_6.1>bit.exe
10111100
```

- This program demonstrates bit manipulation by displaying the binary representation of the value stored in `num` (03Dh). It shifts the bits of `num` to the right one at a time, checking each bit to determine if it is 1 or 0, and outputs the corresponding character ('1' or '0') using interrupt `int 21h`. The result is the binary equivalent of 03Dh (00111101) printed in reverse order (10111100).

Sample Problem B.

1. Modify program bit.asm, replace line number 5 with " **num db 0ah** ".

```
TITLE bit.asm
.model small
.stack 100h

.data
num db 0ah

.code
main proc
    mov ax, @data
    mov ds, ax
    mov bl, num
    mov cx, 8

here:
    shr bl, 1
    jc is_one
    mov dl, 30h
    jmp print

is_one:
    mov dl, 31h

print:
    mov ah, 2
    int 21h
    loop here

exit:
    mov ax, 4c00h
    int 21h

main endp
end main
```

2. Save the program as **bit1.asm**.
3. Assemble and execute the program.

```
C:\HOA_6.1>tasm bit1.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   bit1.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  491k
```

```
C:\HOA_6.1>tlink bit1.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

4. Analyze and record the output in Table 6.2.

Table 6.2 - Output of bit1.asm

```
C:\HOA_6.1>bit1.exe
01010000
```

Did you get the expected output? Why or Why not?

- Yes, I did get the expected output since in the program, for sample problem B, I only modified the initial value that the program will store. So, it outputted the stored values' bits in reverse order.

Sample Problem C.

1. Modify program bit.asm, replace line number 12 with "**here: shr bl,1**".

```
TITLE bit.asm
.model small
.stack 100h

.data
num db 03Dh

.code
main proc
    mov ax, @data
    mov ds, ax
    mov bl, num
    mov cx, 8

here:
    shr bl, 1
    jc is_one
    mov dl, 30h
    jmp print

is_one:
    mov dl, 31h

print:
    mov ah, 2
    int 21h
    loop here

exit:
    mov ax, 4c00h
    int 21h

main endp
end main
```

2. Save the program as **bit2.asm**.
3. Assemble and execute the program.

```
C:\HOA_6.1>tasm bit2.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   bit2.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  491k
```

```
C:\HOA_6.1>tlink bit2.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

4. Analyze and record the output in Table 6.3.

Table 6.3 - Output of bit2.asm

```
C:\HOA_6.1>bit2.exe
10111100
```

Did you get the expected output? Why or Why not?

- Yes, I did get the same output as bit.asm since both bit.asm and bit2.asm contained identical code which shifts the bits of the value stored in 'num' to the right

Sample Problem D.

1. Modify program bit.asm, replace line number 12 with "**here: rcr bl,1**".

```
TITLE bit.asm
.model small
.stack 100h

.data
num db 03Dh

.code
main proc
    mov ax, @data
    mov ds, ax
    mov bl, num
    mov cx, 8

here:
    rcr bl, 1
    jc is_one
    mov dl, 30h
    jmp print

is_one:
    mov dl, 31h

print:
    mov ah, 2
    int 21h
    loop here

exit:
    mov ax, 4c00h
    int 21h

main endp
end main
```

2. Save the program as **bit4.asm**.
3. Assemble and execute the program.

```
C:\HOA_6.1>tasm bit4.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   bit4.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  491k
```

```
C:\HOA_6.1>tlink bit4.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

4. Analyze and record the output in Table 6.4.

Table 6.4 - Output of bit4.asm

```
C:\HOA_6.1>bit4.exe
10111100
```

Did you get the expected output? Why or Why not?

- Yes, This is the expected result because the `rcr` instruction rotates the bits of `bl` (initially `03Dh`) to the right, placing the least significant bit (LSB) into the carry flag. The program checks the carry flag to determine if the LSB is `1` or `0` and outputs the corresponding ASCII character (`'1'` or `'0'`).

B. Supplementary Activity: Output(s) and Observation(s)

1. Modify bit.asm so that it uses 64-bit value stored at NUM. The 64-bit number should be printed from the most significant bit to the least significant bit.

```
TITLE supp1.asm
.model small
.stack 100h

.data
; The 64-bit number is split into 4 16-bit segments
num dw 0F0Fh, 0F0Fh, 0F0Fh, 0F0Fh ;

.code
main proc
    mov ax, @data
    mov ds, ax
    mov cx, 64

here:
    mov ax, num+6
    shl ax, 1
    mov num+6, ax

    mov ax, num+4
    rcl ax, 1
    mov num+4, ax

    mov ax, num+2
    rcl ax, 1
    mov num+2, ax

    mov ax, num
    rcl ax, 1
    mov num, ax

    jc is_one
    mov dl, 30h
    jmp print

is_one:
    mov dl, 31h

print:
    mov ah, 2
    int 21h
    loop here

exit:
    mov ax, 4c00h
    int 21h

main endp
end main
```

```
C:\HOA_6.1>tasm supp1.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   supp1.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  491k


C:\HOA_6.1>tlink supp1.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

```
C:\HOA_6.1>supp1.exe
0000111100001111000011110000111100001111000011110000111100001111
```

- This program performs bit manipulation on a 64-bit number stored as four 16-bit segments (`num`). It repeatedly shifts the bits of the number left, one bit at a time, and checks if the carry flag is set to determine if the shifted-out bit is a `1` or `0`, then prints `1` or `0` accordingly. The process loops 64 times to handle all bits, and the program ends after printing the entire binary representation.

2. Write an assembly program that will clear the most significant nibble, set the least significant nibble and retain the values of the rest of the bits of AX.

```
TITLE supp2.asm
.model small
.stack 100h

.code
main proc
    mov ax, 0ABCDh   ; Load a 16-bit value into AX

    ; Clear the most significant nibble
    and ax, 0FFFh

    ; Set the least significant nibble
    or ax, 000Fh

    ; Preserve AX by copying it to BX
    mov bx, ax

    ; Display the result in binary
    mov cx, 16       ; Set loop counter for 16 bits

display_loop:
    shl bx, 1        ; Shift BX left to bring the MSB into the carry flag
    jc print_one     ; If carry flag is set, print '1'
    mov dl, '0'      ; Otherwise, print '0'
    jmp print_char

print_one:
    mov dl, '1'

print_char:
    mov ah, 2        ; DOS interrupt to display a character
    int 21h
    loop display_loop ; Repeat for all 16 bits

    ; Exit program
    mov ax, 4c00h
    int 21h

main endp
end main
```

```
C:\HOA_6.1>tasm supp2.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   supp2.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  492k


C:\HOA_6.1>tlink supp2.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

```
C:\HOA_6.1>supp2.exe
0000101111001111
```

- The program takes the 16-bit value `0ABCDh`, clears its most significant nibble, sets its least significant nibble, and then displays the resulting binary representation (`1011111111111111`) bit by bit. It outputs the binary digits sequentially as characters on the screen.

3. Make a program that will implement the following 10*ax = 8*ax + 2*ax using shift or rotate instructions.

```
TITLE supp3.asm
.model small
.stack 100h

.code
main proc
    ; Initialize AX with a test value
    mov ax, 0Ah

    ; Save the original value
    mov cx, ax

    ; Calculate 8*ax (shift left by 3)
    shl ax, 1
    shl ax, 1
    shl ax, 1

    ; Calculate 2*ax (shift left by 1)
    mov dx, cx
    shl dx, 1

    ; Add 8*ax + 2*ax to get 10*ax
    add ax, dx

    ; Print the value in AX
    call print_number

    ; Exit program
    mov ah, 4Ch
    int 21h

; Subroutine to print a number in AX
print_number proc
    xor cx, cx
    mov bx, 10

convert_loop:
    xor dx, dx
    div bx
    push dx
    inc cx
    test ax, ax
    jnz convert_loop

print_digits:
    pop dx
    add dl, '0'
    mov ah, 02h
    int 21h
    loop print_digits

    ret
print_number endp

main endp
end main
```

```
C:\HOA_6.1>tasm supp3.asm
Turbo Assembler  Version 2.0  Copyright (c) 1988, 1990 Borland International

Assembling file:   supp3.asm
Error messages:    None
Warning messages:  None
Passes:            1
Remaining memory:  492k


C:\HOA_6.1>tlink supp3.obj
Turbo Link  Version 3.0 Copyright (c) 1987, 1990 Borland International
```

```
C:\HOA_6.1>supp3.exe
100
```

- The program calculates 10 times the initial value of `AX` (which is `0Ah` or 10 in decimal), resulting in `64h` (100 in decimal). It then prints this result (`100`) to the screen using the `print_number` subroutine.

C. Conclusion & Lessons Learned

- In conclusion to the activity, I was able to manipulate bits of data in dosbox using the shift and rotate instructions and also compare the different bit manipulation instructions. In the supplementary activities, I was able to create a program using the bit manipulation instructions such as creating a multiplication instruction. Bit manipulation is one of the advantages of assembly language over high level languages.