Computer Organization & Assembly Language

Lab 10

Topics:

1. Rotate and Shift keywords
2. Nested Procedures (uses keyword)
3. Macros

1. Rotate and shift:

Mnemonic	Function
SHL, SHR	Logical shift left, right byte or word, by 1 or CL
SAL, SAR	Arithmetic shift left, right byte or word, by 1 or CL
ROL, ROR	Rotate left, right byte or word, by 1 or CL
RCL, RCR	Rotate left, right through carry byte or word, by 1 or CL

- SHL operand1, operand2
  - Example
  - MOV AL, 11100000b
  - SHL AL, 1 ; AL = 11000000b
- SHR operand1, operand2
  - Example
  - MOV AL, 00000111b
  - SHR AL, 1 ; AL = 00000011b
- ROL operand1, operand2
  - shift all bits left, the bit that goes off is inserted to the right-most position.
  - Example
  - MOV AL, 1Ch ; AL = 10011100b
  - ROL AL, 1 ; AL = 00111001b
- ROR operand1, operand2
  - shift all bits right, the bit that goes off is inserted to the left-most position
  - Example

- MOV AL, 1Ch ; AL = 00011100b
- ROR AL, 1 ; AL = 00001110b

2. **Nested Procedures (uses keyword):**
   Procedures can also be used in a nested fashion. Any procedure can be called inside any other procedure. In this way the stack will save all the values of instruction pointers and are retrieved in the way of last in first out.
   Keyword uses can be used if a register is to be used inside a procedure.
   Example:

```
Array_Sum proc uses si, cx
<instructions>
ret
Array_Sum endp
```

3. **Macros:**
   - Macros are just like procedures, but they exist only until your code is compiled, after compilation all macros are replaced with real instructions.
   - Macro Definition
     - name MACRO [parameters,...]
     - <instructions>
     - ENDM

   Using Macros:

   - When you want to use a macro, you can just type its name. For example:
     - MyMacro
   - Macro is expanded directly in program's code. So if you use the same macro 100 times, the compiler expands the macro 100 times, making the output executable file larger and larger, each time all instructions of a macro are inserted.

   Passing Arguments to Macro:

   - To pass parameters to macro, you can just type them after the macro name. For example:
     - MyMacro 1, 2, 3
   - To mark the end of the macro ENDM directive is enough

   Example:

- Unlike procedures, macros should be defined above the code that uses it.
- For Example
    - .code
    - mymacro macro p1,p2,p3
    - mov ax,p1
    - mov bx,p2
    - mov cx,p3
    - endm
    - main proc
    - mymacro 1,2,3
    - mov ah,4ch
    - int 21h
    - main endp
    - end

Defining Macros in Separate file:

- To define Macros in Separate file;
    - Open your assembler Source Directory
        - C:\masm615\include\
    - Create a File named “mymacros.inc”
    - Write your Macro in this file and save. Make sure your file have extension .inc
    - Include this file in your source program (*.asm), by writing below line on top of your code
        - include mymacros.asm
    - Compile your Code.

mymacros.inc

```
mymacro macro p1,p2,p3
    mov ax,p1
    mov bx,p2
    mov cx,p3
endm
```

MUS.ASM

```
include \include\mymacros.inc
.model small
.stack
.data
.code
start:
main proc
mymacro 1,2,3
mov ah,4ch
int 21h
main endp
end start
end
```

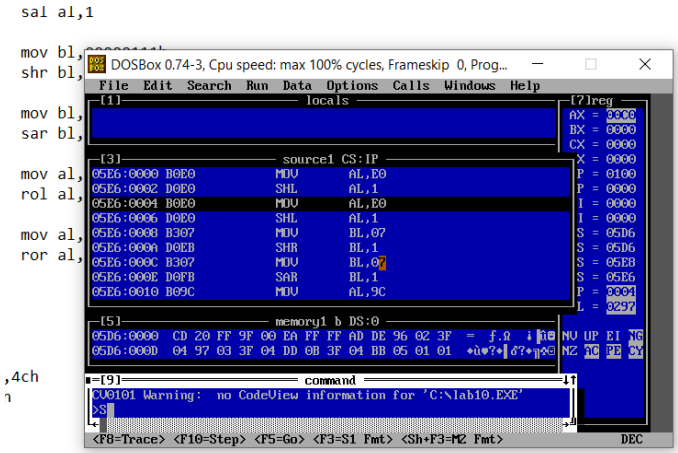
Tasks:

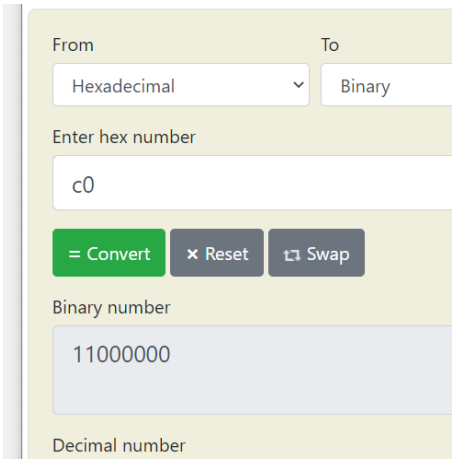
1. Use the keywords of rotate and shift and find out what is the difference between the logic and arithmetic rotate and logic and arithmetic shift.

```
.model small
.stack 100h

.data
.code

        mov al,11100000b
        shl al,1

        mov al,11100000b
        sal al,1

        mov bl,00000111b
        shr bl,1
```

```
mov bl,00000111b
sar bl,1

mov al,10011100b
rol al,1

mov al,00011100b
ror al,1
```

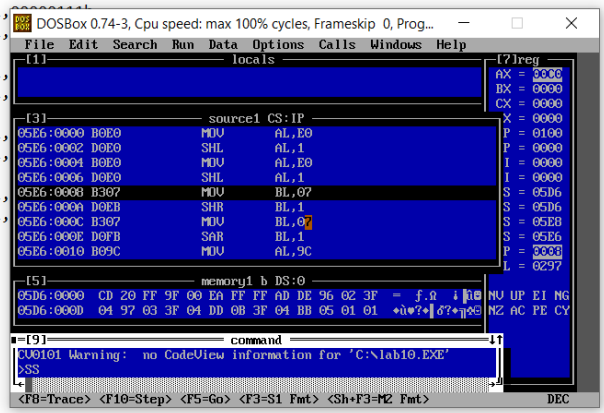
```
mov ah,4ch
int 21h

end
```

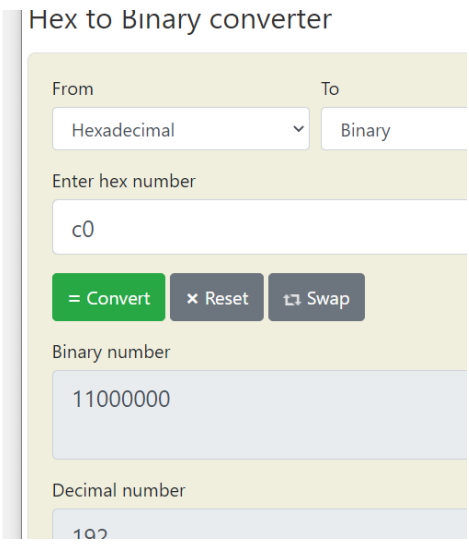




SAL

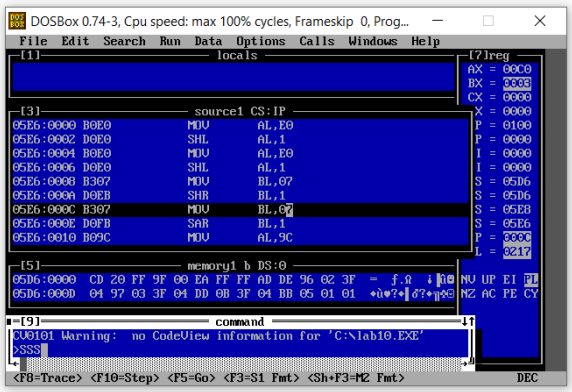
```
mov al,11100000b
sal al,1

mov bl,
shr bl,

mov bl,
sar bl,

mov al,
rol al,

mov al,
ror al,

ch
```

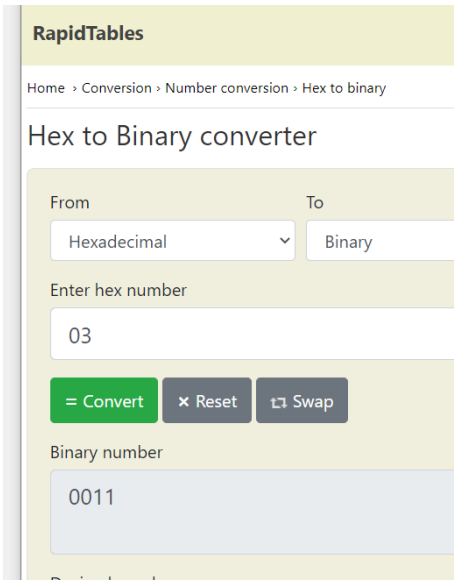




SHR

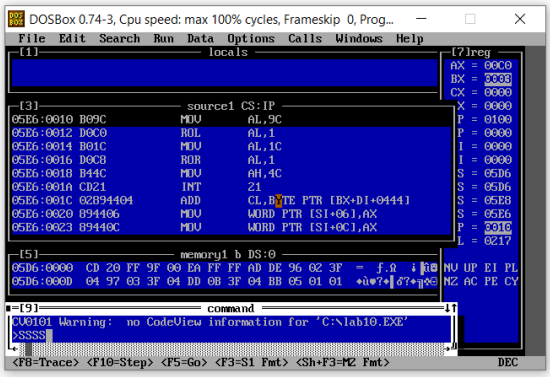
```
ata
ode

mov al,11100000b
shl al,1

mov al,11100000b
sal al,1

mov bl,00000111b
shr bl,1

mov bl,00000111b
sar bl,1

mov al,10011100b
rol al,1

mov al,00011100b
ror al,1

v ah,4ch
t 21h

d
```

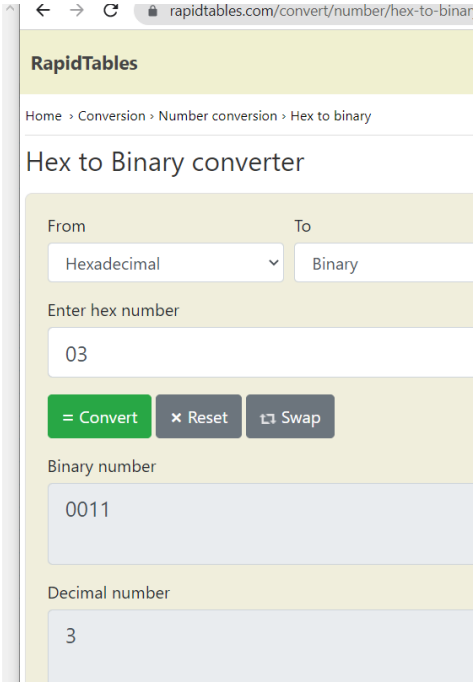




SAR

```
model small
stack 100h

data
code

        mov al,11100000b
        shl al,1

        mov al,11100000b
        sal al,1

        mov bl,00000111b
        shr bl,1

        mov bl,00000111b
        sar bl,1

        mov al,10011100b
        rol al,1

        mov al,00011100b
        ror al,1
```

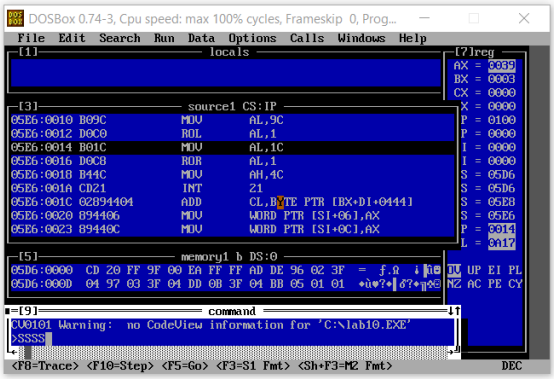


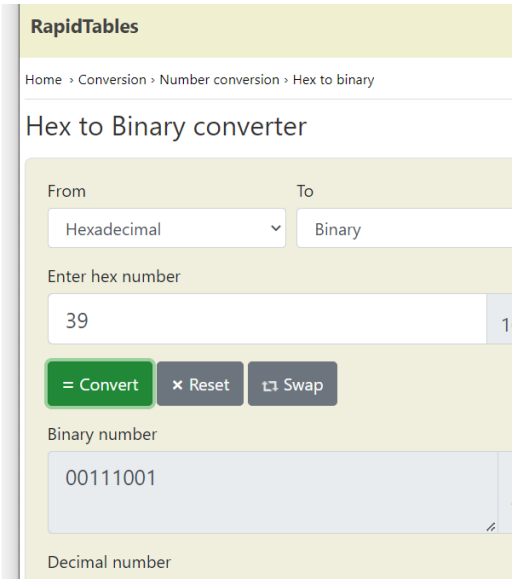



```
ov ah,4ch
nt 21h

nd
```

ROL

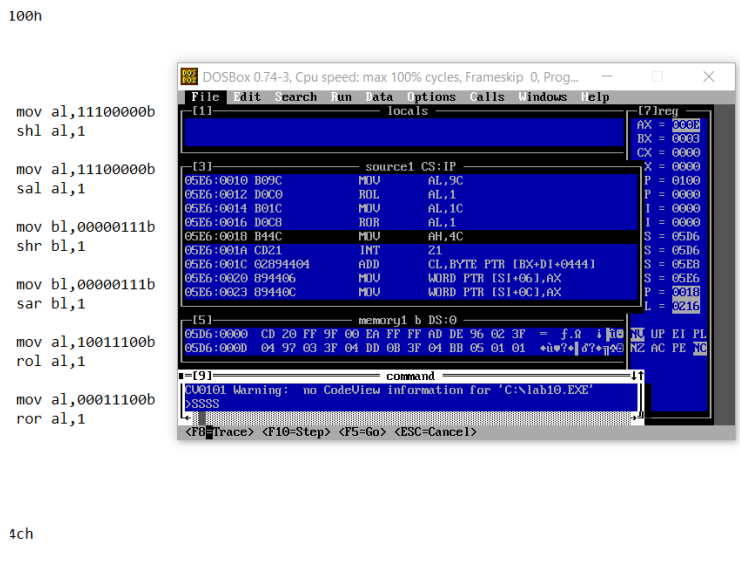
```
mov al,11100000b
shl al,1

mov al,11100000b
sal al,1

mov bl,00000111b
shr bl,1

mov bl,00000111b
sar bl,1

mov al,10011100b
rol al,1

mov al,00011100b
ror al,1
```

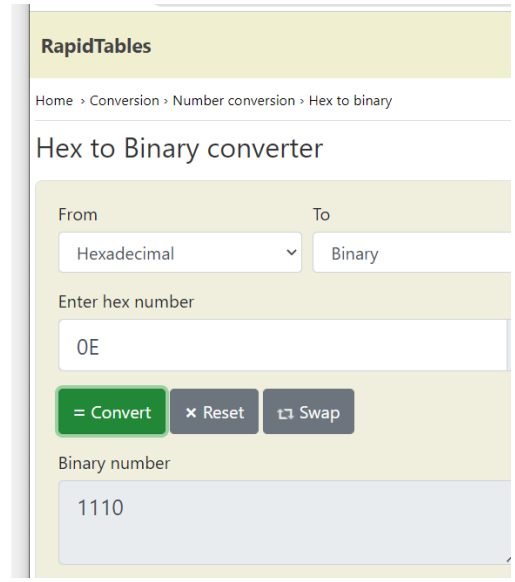





```
,4ch
```

ROR





2. Program a calculator using nested procedures and macros (e.g., for input and output and wherever applicable). The calculator should perform all the basic arithmetic operations.

```
.model small
.stack 100h
.data

        val1 db 0
        val2 db 0
        oper db 0
        result db 0

.code


input macro

        mov ah,01h
        int 21h

        sub al,48
        mov val1,al

        mov ah,01h
        int 21h
```

```
        sub al,48
        mov val2,al

        mov ah,01h
        int 21h
        mov oper,al

endm

output macro
        mov dl,result

        add dl,48
        mov ah,02h
        int 21h
endm


main proc

        input

        cmp oper,2bh
        je addd

        cmp oper,42
        je mulll

        cmp oper,2dh
        je subbb

        cmp oper,47d
        je divvv

        addd:
        call addition
        jmp exit

        subbb:
        call subtraction
        jmp exit

        mulll:
```

```
        call multiplication
        jmp exit

        divvv:
        call division
        jmp exit

main endp

addition proc

        mov bl,val1
        mov cl,val2

        add bl,cl

        mov result,bl
        output

        ret

addition endp

subtraction proc

        mov bl,val1
        mov cl,val2

        sub bl,cl

        mov result,bl
        output

        ret

subtraction endp

multiplication proc

        mov ax,0
        mov al,val1
        mov bl,val2
```

```
        mul bl

        mov result,al
        output
ret

multiplication endp

division proc
        mov ax,0
        mov al,val1
        mov bl,val2

        div bl

        mov result,al
        output

ret
division endp


exit:

mov ah,4ch
int 21h
end
```

3. Write a macro that takes two arguments from user i.e., the first character and the number of characters user wants to print and the macro when called prints the characters starting from the first character (entered by the user) till number of characters.

```
.model small
.stack 100h
.data

        val1 db 0
        val2 db 0
        oper db 0
        result db 0

.code
```

```
mov ax,@data
mov ds,ax
mov ax,0

input macro

        mov ah,01h
        int 21h

        sub al,48
        mov val1,al

        mov ah,01h
        int 21h
        mov oper,al

endm

output macro a1,b1
        mov al,a1
        mov ah,0
        mov cx,ax
        mov dl,b1
L1:

        mov ah,02h
        int 21h

        inc dl

Loop L1

endm

main proc


        input


        output val1,oper
```

```
mov ah,4ch
int 21h

main endp


end
```

4. Write an assembly language program to take an array as input from user. Take an index and a number from user as input. Find out if that number is found at that index (entered by user). If found print ‘True’ otherwise ‘False’. Perform this task by making procedures and macros, wherever applicable.

```
.model small

.stack 100h

.data


        val1 db 8 dup (0)

        val2 db 0

        val3 dw 0

        result db "FOUND$"

        result1 db "NOT FOUND$"


.code


mov ax,@data

mov ds,ax

mov ax,0



;==================================================
```

```
input macro

mov cx,8

mov si,0

l1:

        mov ah,01h
        int 21h

        sub al,48
        mov val1[si],al
        inc si

Loop l1

endm


;==================================================


inputv macro
        mov ah,01h
        int 21h

        sub al,48
```

```
        mov val2,al

endm



inputi macro
        mov ah,01h
        int 21h

        sub al,48
        mov ah,0
        mov val3,ax

endm

;==================================================



output macro a1,b1
        mov al,a1
        mov ah,0
        mov cx,ax
        mov dl,b1
L1:

        mov ah,02h
        int 21h
```

```
        inc dl

Loop L1

endm


;==================================================


main proc


        input  ;aray

        inputv ;value

        inputi ;index

        mov si,val3

        mov al,val1[si]

        cmp al,val2
        je f

        jmp nf
```

```
f:
        mov dx,offset result
        mov ah,09h
        int 21h
jmp exit


nf:

        mov dx,offset result1
        mov ah,09h
        int 21h


exit:

mov ah,4ch
int 21h

main endp


end
```