Green University in Bangladesh

Dept:CSE

Course Title: Microprocessors & Microcontrollers Lab

Course Code: CSE_304

Submission Date:08.05.2021

Submitted By	Submit To
Name: Md.Abdullah al noman	**Name:** Mr.Abdullah Al arif
ID: 191015145 **Section: Ec** **Depertment: CSE**	**Designation : Lecturer** **Dept .CSE** **GUB**

Green University in Bangladesh

Dept:CSE

Course Title: Microprocessors & Microcontrollers Lab

Course Code: CSE_304

Experiment Name: Arithmetic operations

Submitted By	Submit To
Name: Md.Abdullah al noman	**Name:** Mr.Abdullah Al arif
ID: 191015145 **Section: Ec** **Depertment: CSE**	**Designation : Lecturer** **Dept .CSE** **GUB**

Learning Objective: To add and subtract two 8 bit or 16-bit numbers To perform multiplication and division arithmetic operations over two 8 bit or 16-bit numbers

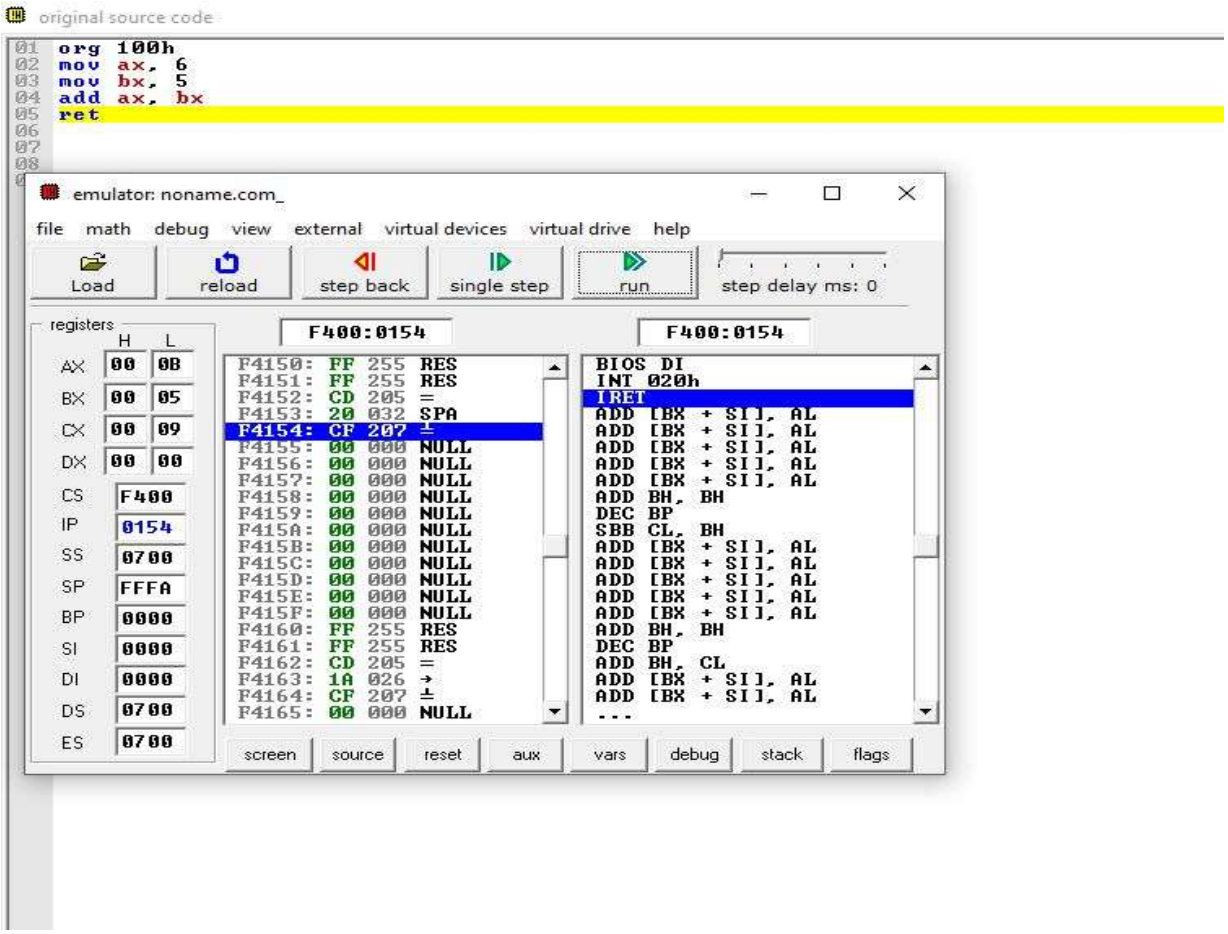
Theory:

We use ADD instruction for addition and SUB instruction for subtraction. ADD instruction adds an immediate data or contents of a memory location specified in the instruction or a register (source) to the contents of another register (destination) or memory location Hence one of the operands is initially moved to AL OR AX. Then using the add instruction, 8 bit or 16-bit addition is performed. The next arithmetic primitive is SUB. As discussed in ADD it permits the same modes of addressing we use MUL instruction and for division we use DIV instruction.MUL instruction multiplies unsigned byte or word by the content of AL. The unsigned byte or word may be in any one of the general-purpose register or memory locations DIV instruction divides an unsigned word or double word by a 16-bit or 8-bit operand. The dividend must be in AX for 16-bit operation and divisor may be specified using any one of the addressing modes except immediate. The result will be in AL (quotient) while AH will contain the remainder. If the result is too big to fit in AL, type 0(divide by zero) interrupt is generated

Addition:

```
org 100h
mov ax, 6
mov bx, 5
add ax, bx
ret
```

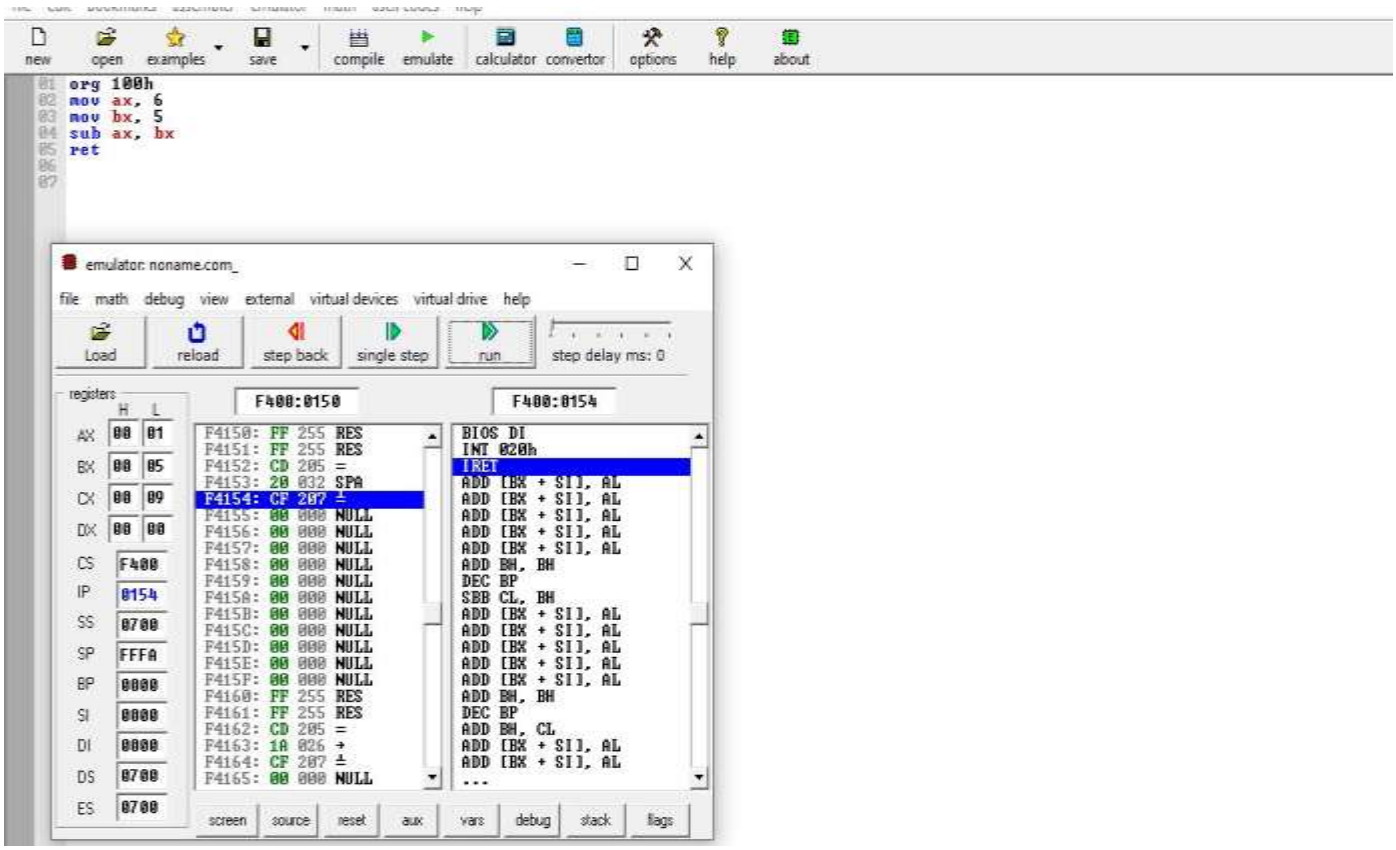
Oootput:



Subtraction:

```
org 100h
mov ax, 6
mov bx, 5
sub ax, bx
ret
```

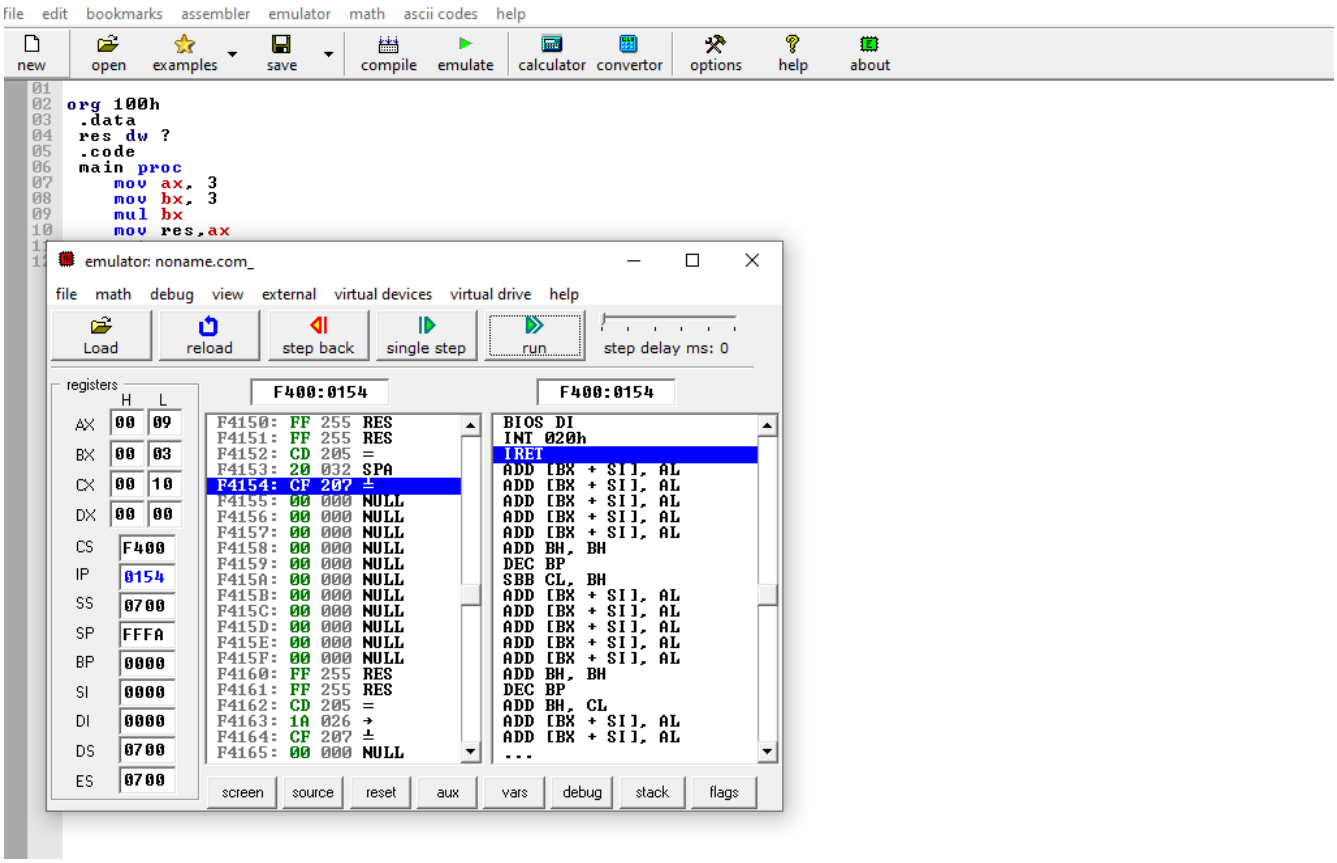
Output:



Multiplication:

```
org 100h
 .data
 res dw ?
 .code
 main proc
    mov ax, 3
    mov bx, 3
    mul bx
    mov res,ax
    ret
```

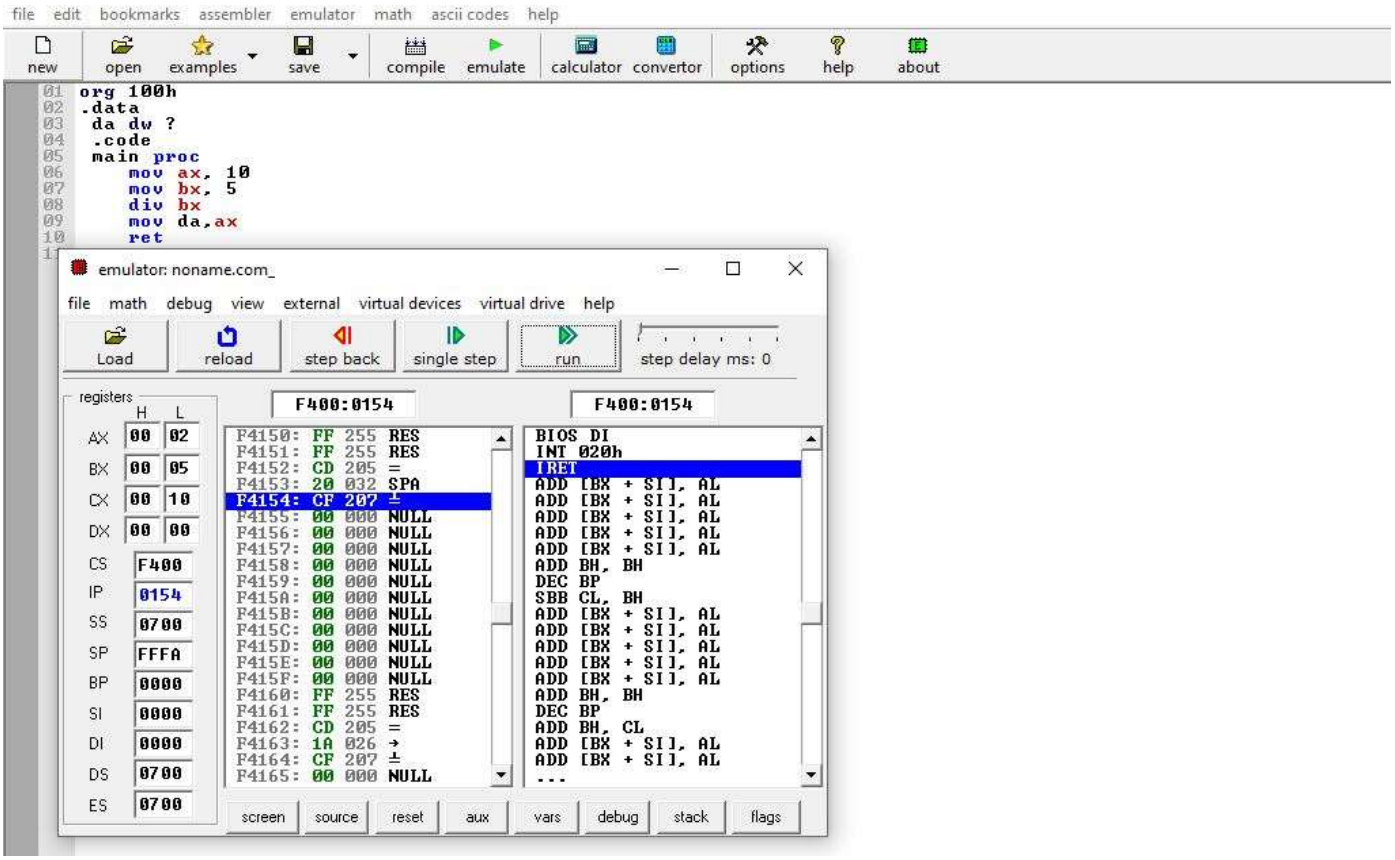
Output:



Division:

```
org 100h
.data
 da dw ?
 .code
 main proc
   mov ax, 10
   mov bx, 5
   div bx
   mov da,ax
   ret
```

Output:



Discussion:

The ADD and SUB instructions are used for performing simple addition/subtraction of binary data in byte, word and doubleword size, i.e., for adding or subtracting 8-bit, 16-bit or 32-bit operands, respectively.. MUL instruction and for division we use DIV instruction.MUL instruction multiplies unsigned byte or word by the content of AL. The unsigned byte or word may be in any one of the general-purpose register or memory locations DIV instruction divides an unsigned word or double word by a 16-bit or 8-bit operand.

Reference: www.tutorialspoint.com

Green University in Bangladesh

Dept:CSE

Course Title: Microprocessors & Microcontrollers Lab

Course Code: CSE_304

Experiment Name: Jump in 8086

Submission Date:08.05.2021

Submitted By	Submit To
Name: Md.Abdullah al noman	**Name:** Mr.Abdullah Al arif
ID: 191015145 **Section: Ec** **Depertment: CSE**	**Designation : Lecturer** **Dept .CSE** **GUB**

Learning objective: Implementing jump in emu 8086 will give us a understanding of how jump work and control the flow.

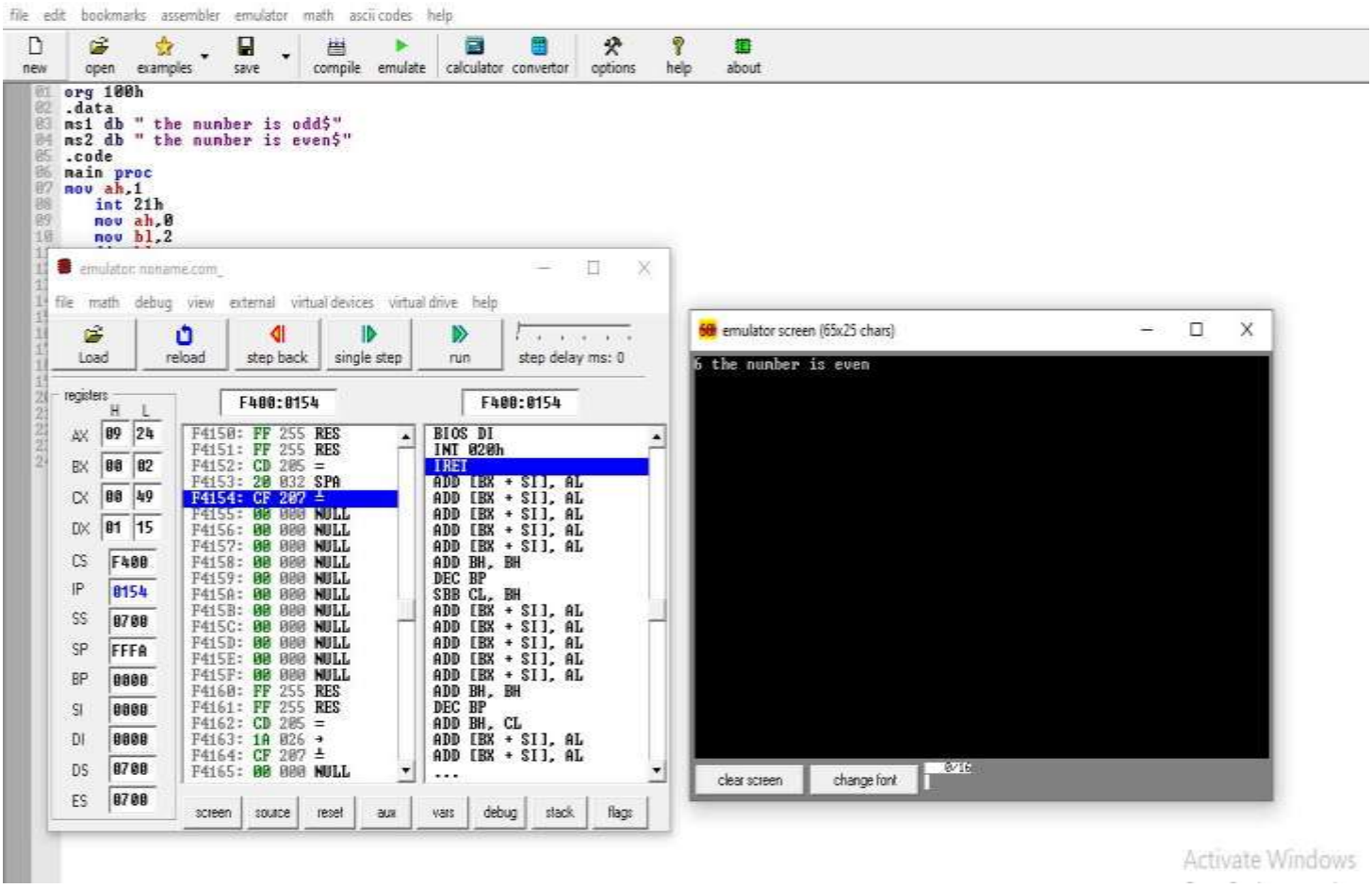
Theory: Jump instructions transfers the program sequence to the memory address given in the operational based on specific flags. There can be 2types of jump instructions, Conditional and unconditional jump.

jump:

```
jump:
org 100h
.data
ms1 db " the number is odd$"
ms2 db " the number is even$"
.code
main proc
mov ah,1
  int 21h
  mov ah,0
  mov bl,2
  div bl
  cmp ah,0
 je s1
mov ah,9
lea dx,ms1
```

```
 int 21h
jmp s2
  s1:
mov ah,9
 lea dx,ms2
 int 21h
  s2:
  ret
```

Output:



Discussion: jump Instructions are used for changing the flow of execution of instructions in the processor. If we want jump to any instruction in between the code, then this can be achieved by these instruction

Reference: www.tutorialspoint.com

Green University in Bangladesh

Dept:CSE

Course Title: Microprocessors & Microcontrollers Lab

Course Code: CSE_304

Experiment Name: Loop & Array

Submission Date:08.052021

Submitted By	Submit To
Name: Md.Abdullah al noman	**Name:** Mr.Abdullah Al arif
ID: 191015145 **Section: Ec** **Depertment: CSE**	**Designation : Lecturer** **Dept .CSE** **GUB**

Learning Objective: The JMP instruction can be used for implementing loops. For example, the following code snippet can be used for executing the loop-body 10 times MOV CL, 10 L1: DEC CL JNZ L1

Theory:

The processor instruction set however includes a group of loop instructions for implementing iteration. The basic LOOP instruction has the following syntax: LOOP label Where, label is the target label that identifies the target instruction as in the jump instructions. The LOOP instruction assumes that the ECX register contains the loop count. When the loop instruction is executed, the ECX register is decremented and the control jumps to the target label, until the ECX register value, the counter reaches the value zero.

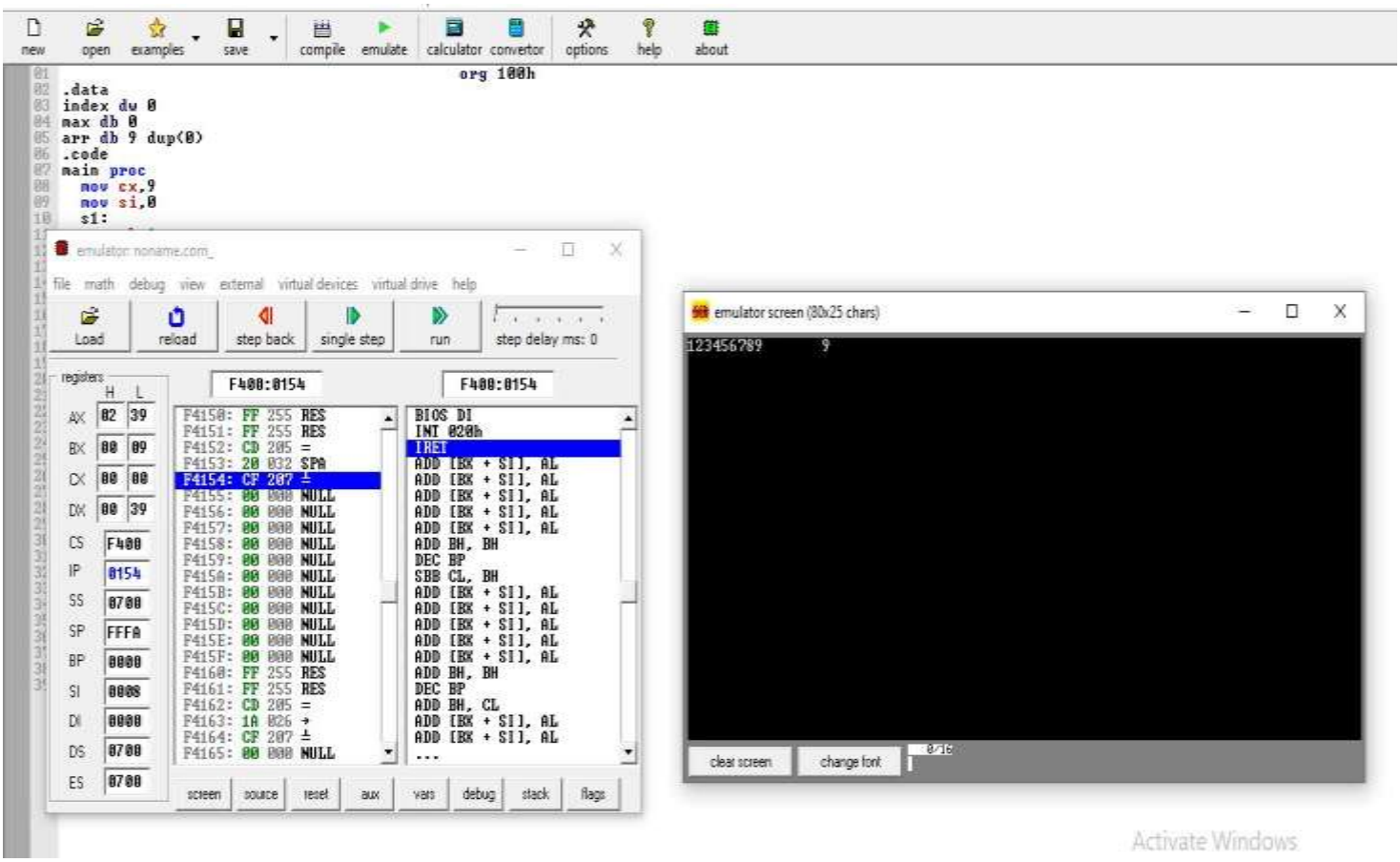
Number Print:

Code loop & Array:

```
org 100h
.data
index dw 0
max db 0
arr db 9 dup(0)
.code
main proc
  mov cx,9
  mov si,0
  s1:
  mov ah,1
  int 21h
  sub al,48
  mov arr[si],al
```

```
add si,1
loop s1

mov ah,2
mov dl,9
int 21h
mov cx,9
s2:
mov ah,2
mov bl,max
mov si,index
cmp bl,arr[si]
jnl s3
mov bl,arr[si]
mov max,bl

s3:
add index,1
loop s2
mov ah,2
mov dl,max
add dl,48
int 21h
ret
```

Output:



Discussion: JMP instruction can be used for implementing loops. For example, the following code snippet can be used for executing the loop-body 10 times.

Reference: www.tutorialspoint.com

Green University in Bangladesh

Dept:CSE

Course Title: Microprocessors & Microcontrollers Lab

Course Code: CSE_304

Experiment Name: procedure & marco

Submission Date:8.05.2021

Submitted By	Submit To
Name: Md.Abdullah al noman	**Name:** Mr.Abdullah Al arif
ID: 191015145 **Section: Ec** **Depertment: CSE**	**Designation : Lecturer** **Dept .CSE** **GUB**

Learning objectives: from this experiment we will understand their work. procedure is the process and marco is like a processing sustyem we can store and use later.

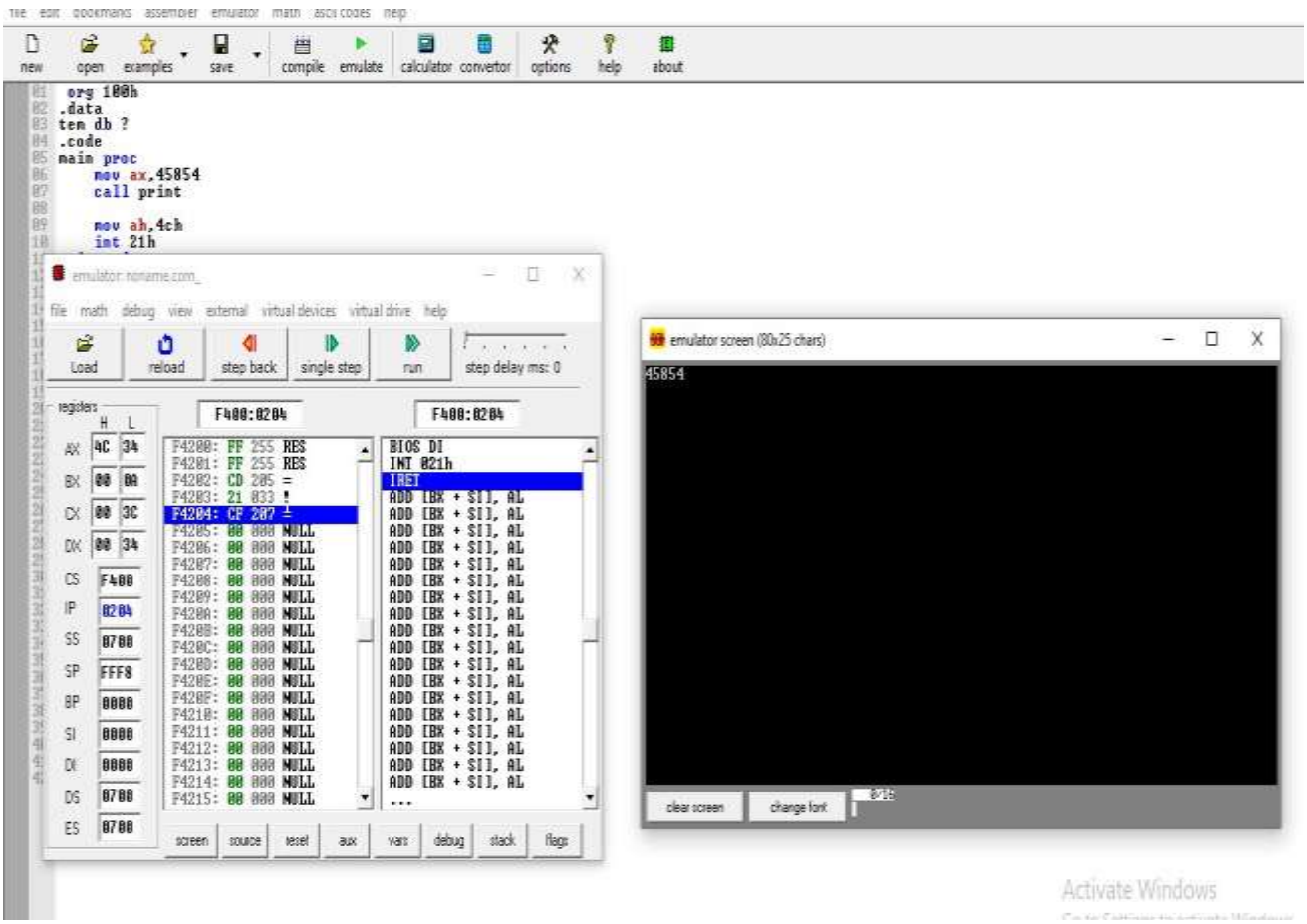
Theory: Macro definition contains a set of instruction to support modular programming. Procedure contains a set of instructions which can be called repetitively which can perform a specific task.

Procedure:

```
org 100h
.data
tem db ?
.code
main proc
   mov ax,45854
   call print

   mov ah,4ch
   int 21h
main endp


nprint proc
   mov ah,2
   add dl,48
   int 21h
```

```
   ret
nprint endp


print proc
     mov bx,10
     mov si,0
     s1:
     mov dx,0
      div bx
       mov tem[si],dl
         add si,1
         cmp ax,0
         jne s1
         s2:

    sub si,1
    mov dl,tem[si]
    call nprint
    cmp si,0
    jne s2
    ret
print endp
```

Output:

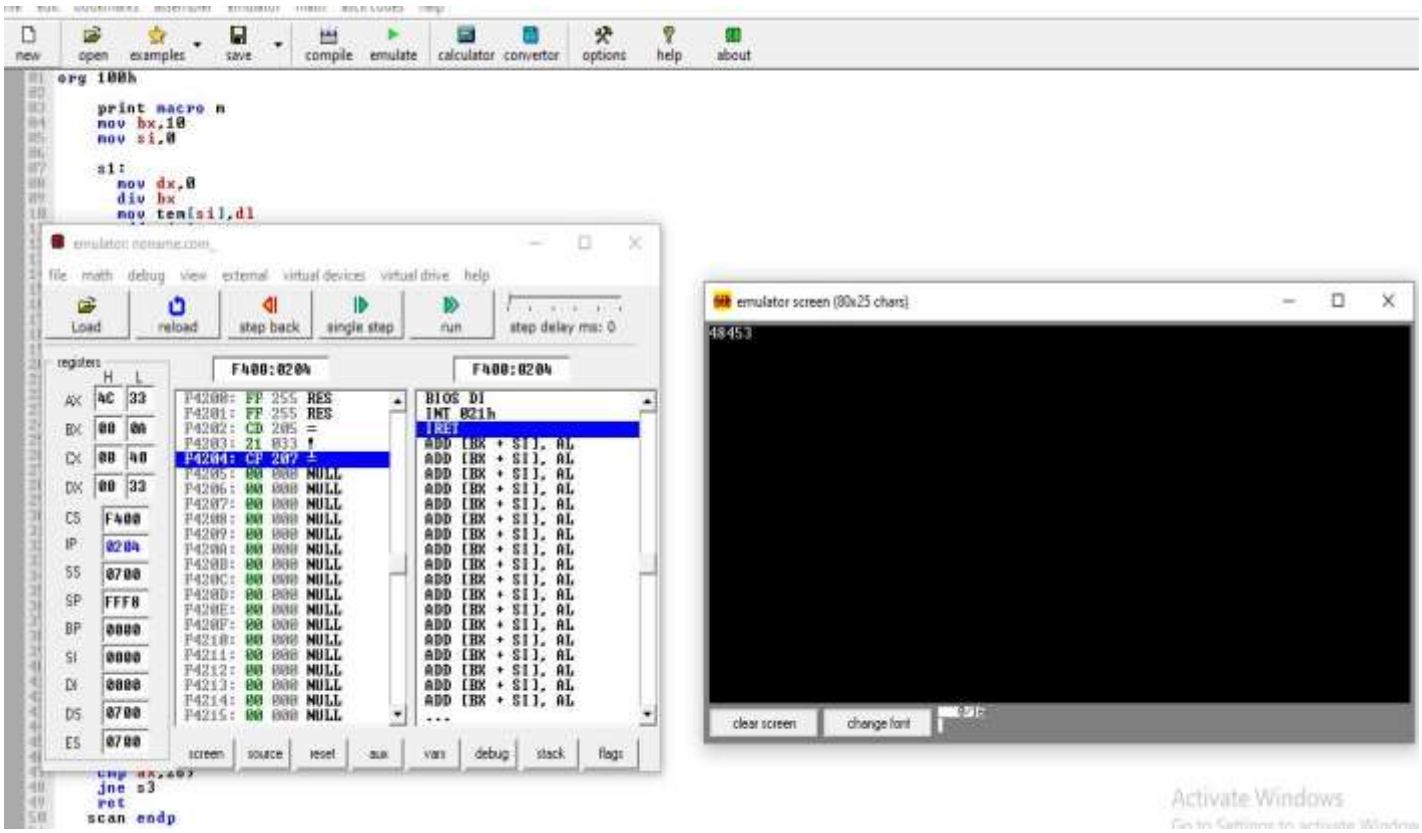

file edit bookmarks assembler emulator math ascii codes help
new
open
examples
save
compile
emulate
calculator
convertor
options
help
about
org 100h
.data
ten db ?
.code
main proc
mov ax,45854
call print
mov ah,4ch
int 21h
emulator: noname.com_
file math debug view external virtual devices virtual drive help
Load
reload
step back
single step
run
step delay ms: 0
registers
H L
AX 4C 34
BX 00 00
CX 00 3C
DX 00 34
CS F400
IP 0204
SS 0700
SP FFF8
BP 0000
SI 0000
DI 0000
DS 0700
ES 0700
F400:0204
F400:0204
F4200: FF 255 RES
F4201: FF 255 RES
F4202: CD 205 =
F4203: 21 033 !
F4204: CF 207 ╧
BIOS DI
INT 021h
IRET
ADD [BX + SI], AL
screen
source
reset
aux
vars
debug
stack
flags
emulator screen (80x25 chars)
45854
clear screen
change font
Activate Windows

Macro:

```
org 100h

   print macro m
   mov bx,10
   mov si,0

   s1:
     mov dx,0
     div bx
     mov tem[si],dl
     add si,1
     cmp ax,0
     jne s1

     s2:
     sub si,1
     mov dl,tem[si]
     nprint m
     cmp si,0
     jne s2
     endm

    nprint macro n
    mov ah,2
```

```
 add dl,48
 int 21h
 endm


.data
number dw 5
tem db ?
.code
main proc
 mov ax,48453
 print ax

 mov ah,4ch
 int 21h
 main endp

 scan proc

 s3:
 mov ah,1
 int 21h
 cmp ax,269
 jne s3
 ret
scan endp
```

Output:



Discussion: Procedures or subroutines are very important in assembly language, as the assembly language programs tend to be large in size. Procedures are identified by a name. The macro is invoked by using the macro name along with the necessary parameters. When you need to use some sequence of instructions many times in a program, you can put those instructions in a macro and use it instead of writing the instructions all the time.

Reference: www.tutorialspoint.com