Green University of Bangladesh

Dept. of CSE

Course Title: Microprocessor and Microcontroller Lab

Lab Report

Submitted By:	Submitted To:
Md. Tanvir Ahmed	Mr. Abdullah Al Arif
ID No : 191015102	Designation: Lecturer
Batch No: EC	Dept. of CSE
Semester : 7th	Green University Of Bangladesh

1. Arithmetic operations.

Experiment Name: Program to find the sum of two numbers.

Experiment No: 01

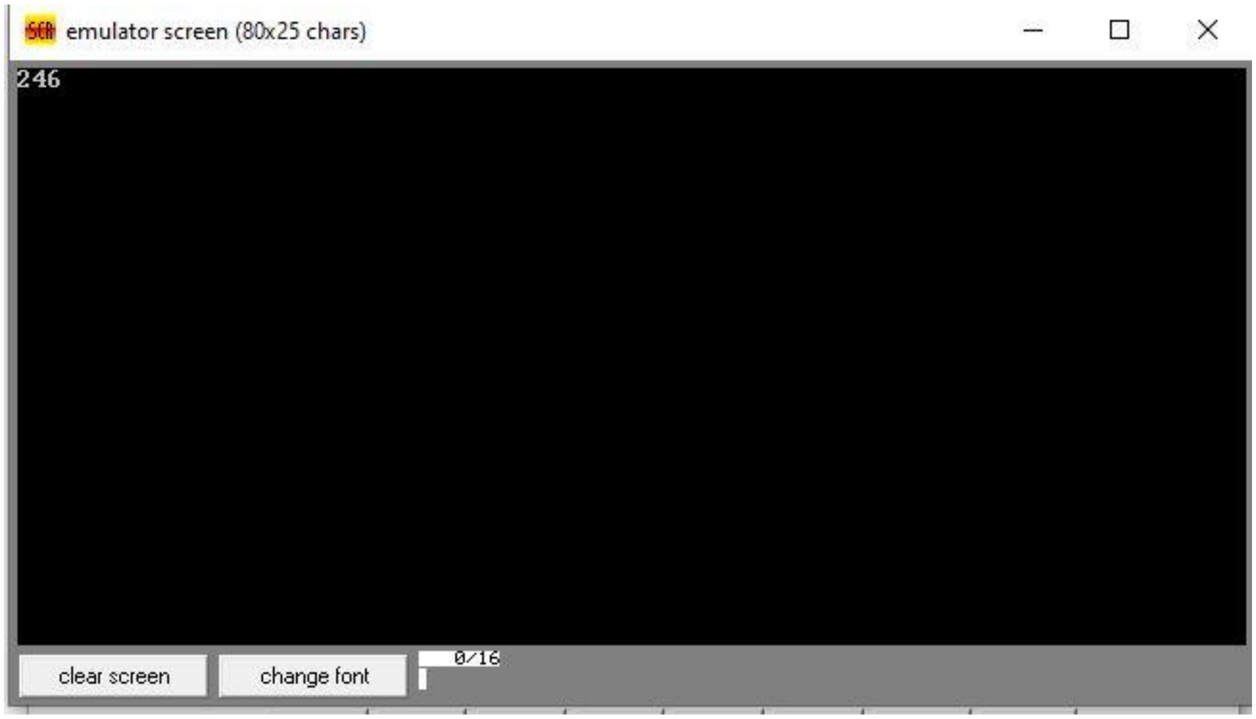
Learning Objective: How to sum two decimal numbers.

Theory: Initialize HL Register pair with address where the first number. Store the number in accumulator. Get the second number. Add the two numbers and store the result in 6. Go back to Monitor

Code:

```
org 100h
.data
num db ?
num2 dw ?
.code
main proc
   mov ah, 1
   int 21h
   sub al,48
   mov num,al
   int 21h
   sub al,48
   add al,num
   mov ah,2
   mov dl,al
   add dl,48
   int 21h
ret
```

Output Screenshot:



```
246
```

Discussion: Two numbers can be sum to this program, but the sum of the two numbers will be displayed from 0 to 9, not above.

Experiment Name: Write a program to subtract two decimal numbers.

Experiment No: 02

Learning Objective: How to subtract two decimal numbers.

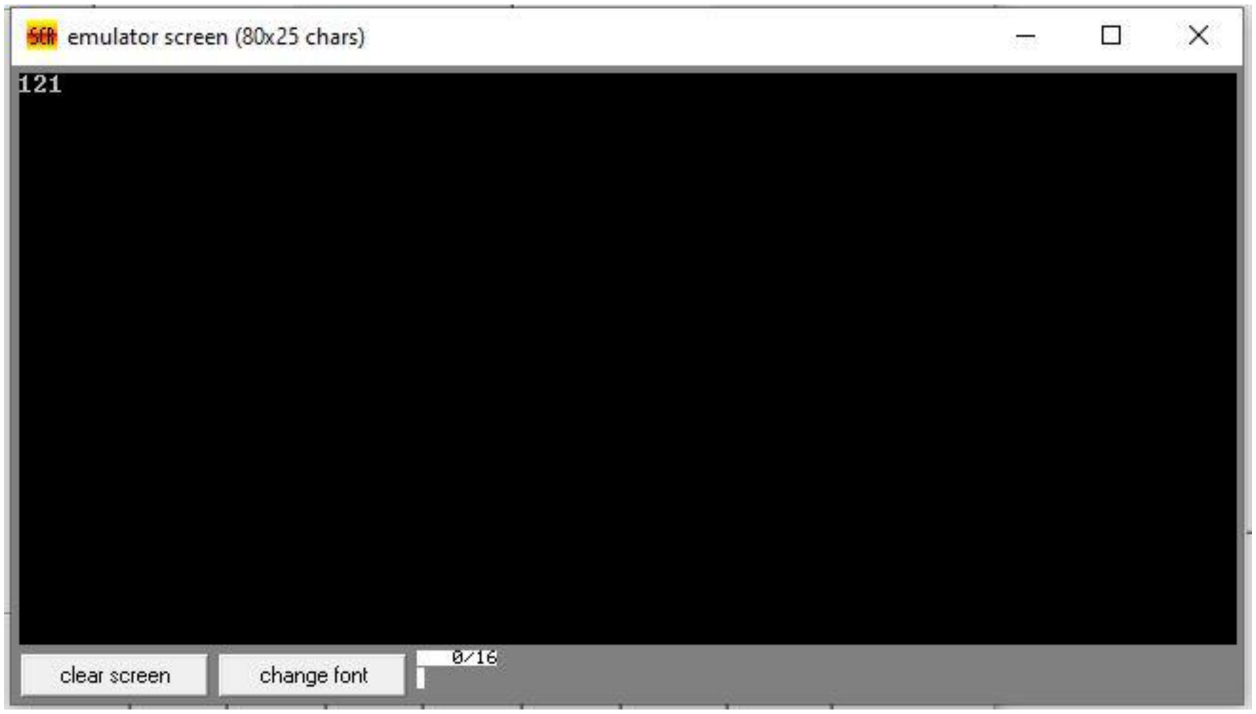
Learning Objective: How to subtract two decimal numbers.

Theory: Initialize HL Register pair with address where the first number is lying. Store the number in accumulator. Get the second number. Subtract second no from acc and store the result in 2. Adjust the decimal Go back to Monitor.

Code:

```
org 100h
.data
num db ?
num2 dw ?
.code
main proc
   mov ah,1
   int 21h
   sub al,48
   mov num,al
   int 21h
   sub al,48
   sub al,num
   mov ah,2
   mov dl,al
   add dl,48
   int 21h
ret
```

Output Screenshot:



```
121
```

2. Jump in emu 8086.

Experiment Name: Program to display the largest number using jump condition.

Experiment No: 03

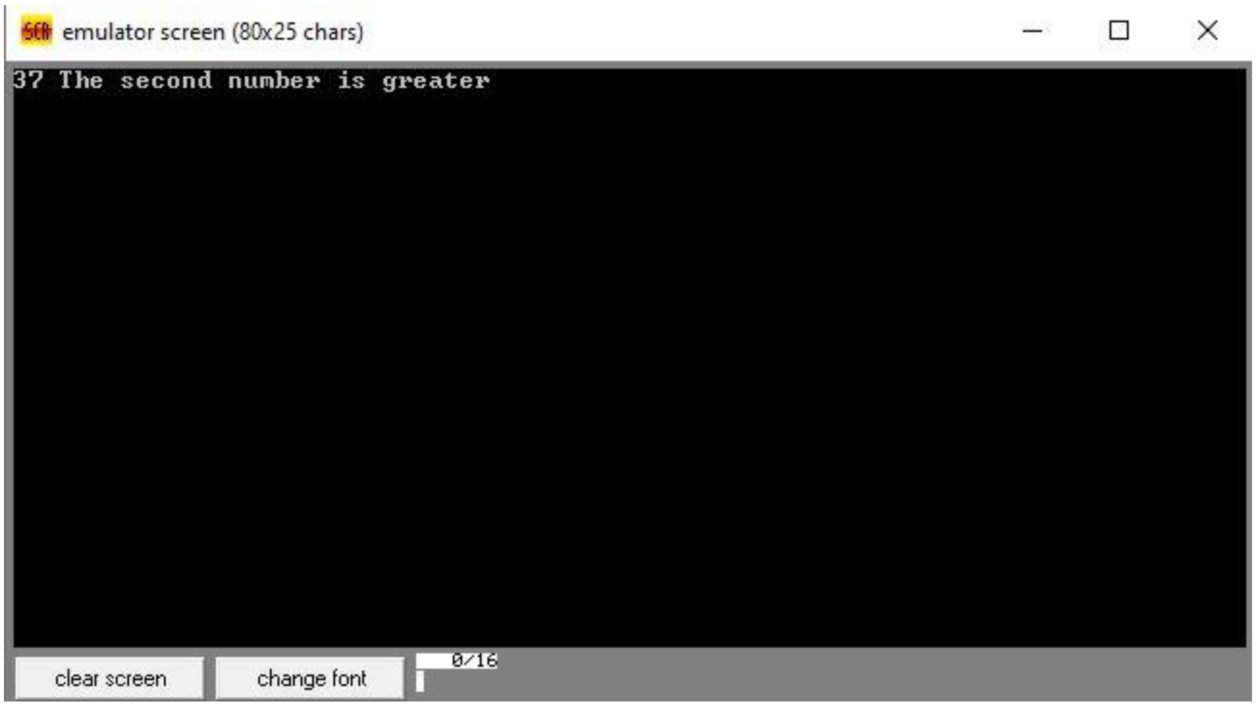
Learning Objective: How to find largest number within two numbers.

Theory: 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 instructions. There are two types of Jump instructions.

Code:

```
org 100h
.data
ms1 db " The first number is greater$"
ms2 db " The second number is greater$"
.code
main proc
    mov ah,1
    int 21h
    mov bl,al
     int 21h
cmp bl,al
jg s1
    mov ah,9
    lea dx, ms2
    int 21h
    jmp s2
s1:
mov ah,9
lea dx, ms1
int 21h
s2 :
ret
```

Output Screenshot:



```
37 The second number is greater
```

Discussion: Using this condition, giving two numbers in the input will show which the largest number.

3. Loop & amp; Array.

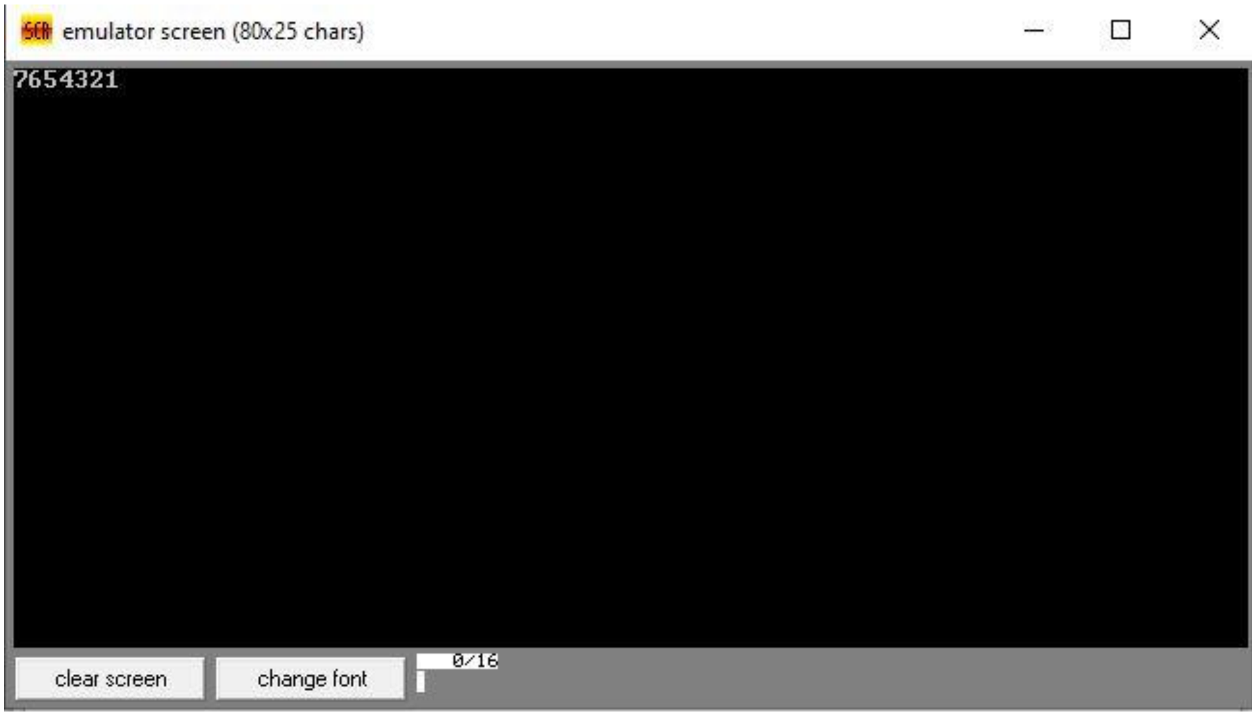
Experiment Name: Display numbers from 1 to 7 in reverse using loop condition.
Experiment No: 04
Learning Objective: How to use loop condition.

Theory: Loop instructions are used to simplify discriminating testing and bouncing portion of the loop. In the above case this portion required two instructions, but in more complicated situation may require more than two instruction.

Code:

```
org 100h
 .data
 .code
 main proc
  mov cx,9
  s1:
  mov ah,2
  mov dl,cl
  add dl,48
  int 21h
  loop s1
  s2:
  s3:
  ret
```

Output Screenshot:

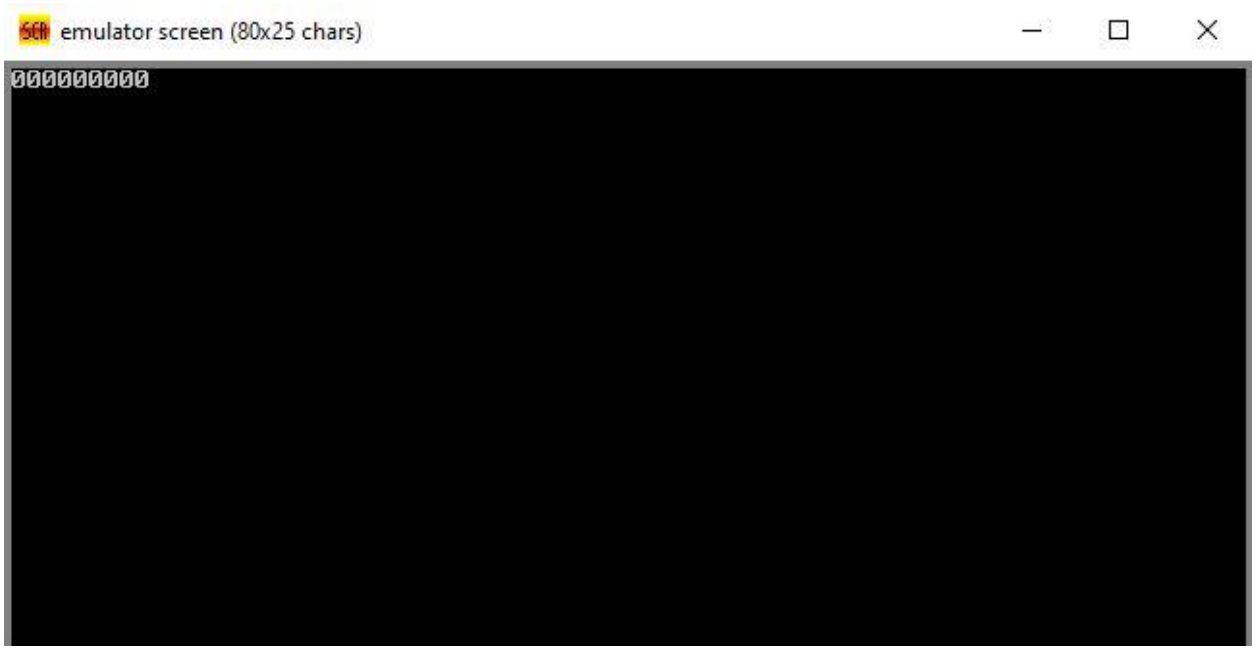


```
7654321
```

Discussion: This program displays numbers from zero to 7 in reverse through the loop.

Experiment Name: Writing a program to display zero 9 times using the array condition.
Experiment No: 05
Learning Objective: How to use array condition.
Theory: Compare value of register AL from data value at next offset, if that data is greater than value of register AL then update value of register AL to that data else no change, and increase offset value for next comparison and decrease count by 1 and continue this till count value of register CX becomes 0.
Store the result to memory address.
Code:
```
org 100h
.data
index dw 0
arr db 9 dup(0)
.code
main proc
     mov cx,9
     s1:
     mov ah,2
     mov si,index
     mov dl,arr[si]
     add dl,48
     int 21h
     add index,1
     loop s1
     s2:
     s3:
    ret
```

Output Screenshot:



```
000000000
```

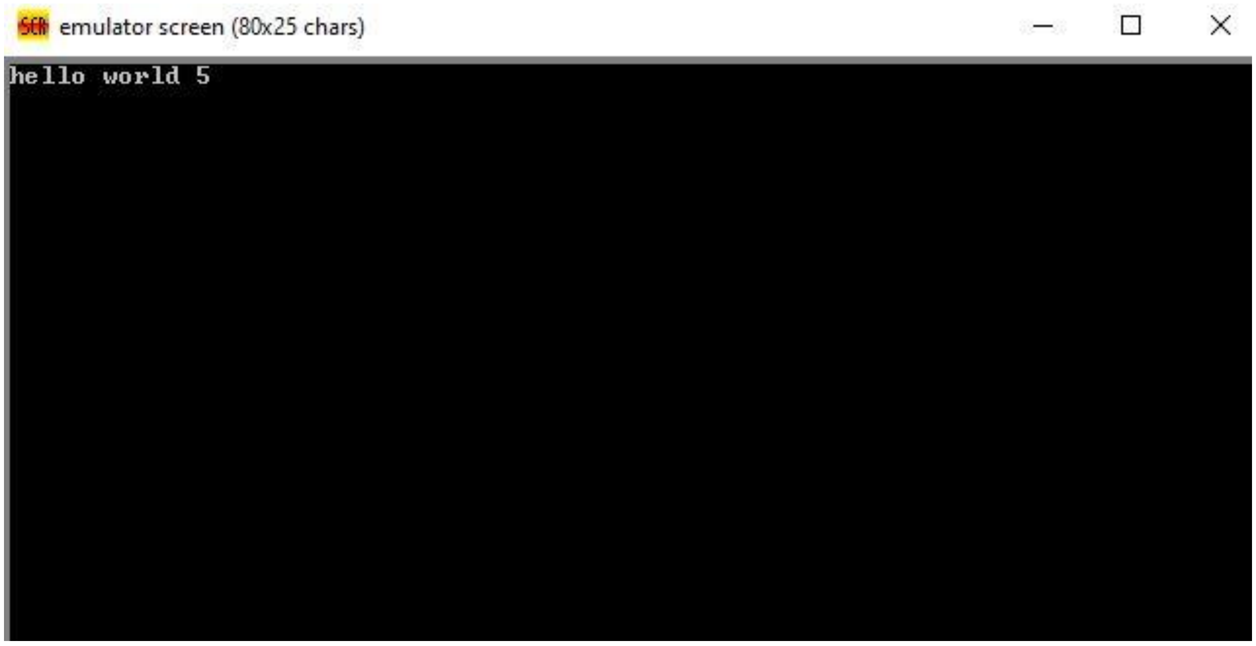
Discussion: When we run the program in this array, whatever number we give as input venue will display zero. We can display another number if we want

Experiment Name: Writing a program to display string and number using the Procedure.
Experiment No: 06
Theory: ORG 100h is a compiler directive, it tells compiler how to handle the source code. This directive is very important when you work with variables. It tells compiler that the executable file will be loaded at the offset of 100h 256 bytes, so compiler should calculate the correct address for all variables when it replaces the variable names with their offsets.

Code:

```
org 100h
  .data
  ms1 db "hello world $"
  .code
  main proc
   lea dx,ms1
    call sprint
    mov dl,5
    call nprint
      lea dx,ms1
    mov ah,4ch
    int 21h
    main endp
  sprint proc
    mov ah,9
    int 21h
    ret
    sprint endp
   nprint proc
    mov ah,2
    add dl,48
    int 21h
    ret
    nprint endp
```

Output Screenshot:



```
hello world 5
```

Discussion:
If I want to have seen first word Hello World, this assembly program prints STRING from .DATA, but in three different lines. So, here we'll be saving 3 different Strings to .DATA, and then print them one by one. From our first Hello World ALP hope you have understood the use of mnemonics to display the string and the interrupt request.

Experiment Name: Write a program to display any number of digits using macro condition.

Experiment No: 07

Theory: Macros are just like procedures, but not really. Macros look like procedures, but they exist only until your code is compiled, after compilation all macros are replaced with real instructions. If you declared a macro and never used it in your code, compiler will simply ignore it A Macro is a set of instructions grouped under a single unit. It is another method for implementing modular programming in the 8086 microprocessors.

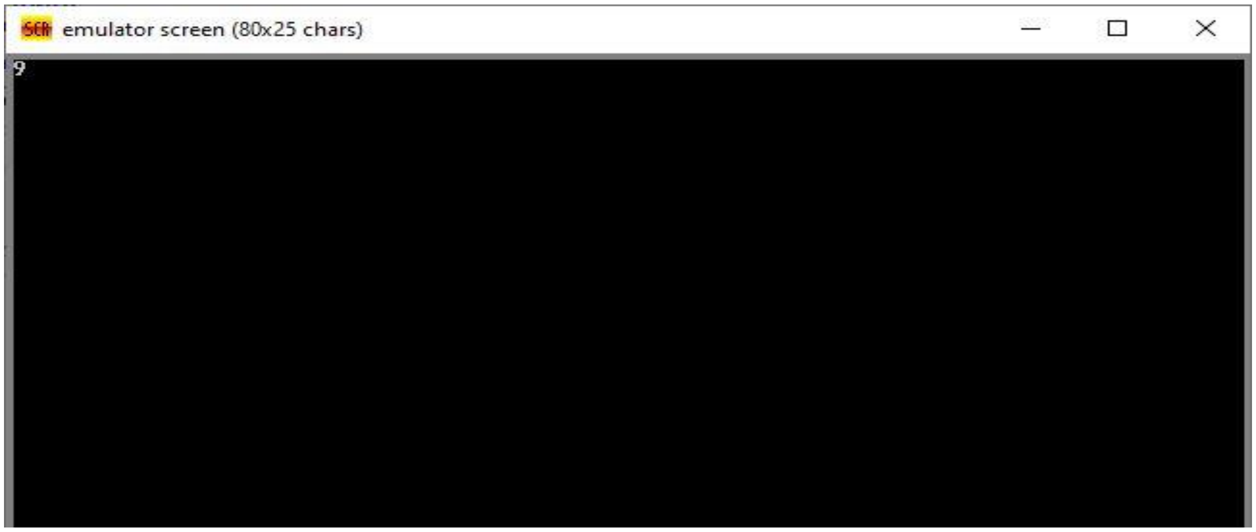
The Macro is different from the Procedure in a way that unlike calling and returning the control as in procedures, the processor generates the code in the program every time whenever and wherever a call to the Macro is made.

Code:

```
org 100h
print macro m
   mov ah,2
   mov dl,m
   add dl,48
   int 21h
endm
.data
number db 9
.code
main proc
   print number

   mov ah, 4ch
   int 21h
   main endp
```

Output Screenshot:



```
9
```

Discussion: A Macro can be defined in a program using the following assembler directives MACRO used after the name of Macro before starting the body of the Macro and ENDM at the end of the Macro) All the instructions that belong to the Macro lie within these two assembler directives. The following is the syntax for defining a Macro in the 8086 Microprocessor.

Reference: Google

Thank You