**Practical - 28**

**Aim:** Write an assembly language program to perform addition of two variables a and b using MACRO.

**Description of instructions used:**

**MOV** - Used to copy the byte or word from the provided source to the provided destination.

**ADD** - Used to add the provided byte to byte/word to word.

**MACRO** - 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.

**Macro definition:**

Name

MACRO [parameters,...]

ENDM

**important facts about macros and procedures**:

♣ Procedure is located at some specific address in memory, and if you use the same procedure 100 times, the CPU will transfer control to this part of the memory. The control will be returned back to the program by RET instruction. The stack is used to keep the return address.

♣ 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.

♣ You should use stack or any general purpose registers to pass parameters to procedure.

♣ 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.

♣ To mark the end of the procedure, you should type the name of the procedure before the ENDP directive.

**Code:**

.model pract28

.data

a dw 5

b dw 6

c dw 1 dup(?)

.code

sum macro a, b

mov ax, a

mov bx, b

add ax, bx

mov c, ax

endm

mov dx, data

mov ds, dx

sum a, b

hlt

**Output:**

