EXPERIMENT - 9

**AIM**: Write Programs using Assembly Language:

1. To implement Macros for calculating Factorial of a number.

2. To study and Implement DOS interrupts. Calculate and display Sum of 2 user entered inputs using DOS interrupt

**Submission Sheet**

| **SAP ID** | **Name of Student** | **Date of Experiment** | **Date of Submission** | **Remarks** |
| --- | --- | --- | --- | --- |
| 60004190057 | Junaid Girkar |  |  |  |

**Theory** :

**Macros** : Writing a macro is another way of ensuring modular programming in assembly language. A macro is a sequence of instructions, assigned by a name and could be used anywhere in the program. In NASM, macros are defined with %macro and %endmacro directives. The macro begins with the %macro directive and ends with the %endmacro directive. The Syntax for macro definition −

| %macro macro\_name number\_of\_params  <macro body> %endmacro |
| --- |

Where, number\_of\_params specifies the number parameters, macro\_name specifies the name of the macro. 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

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. Unlike procedures, macros should be defined above the code that uses it.

The advantage of using Macro is that it avoids the overhead time involved in calling and returning (as in the procedures). Therefore, the execution of Macros is faster as compared to procedures. Another advantage is that there is no need for accessing stack or providing any separate memory to it for storing and returning the address locations while shifting the processor controls in the program.

But it should be noted that every time you call a macro, the assembler of the microprocessor places the entire set of Macro instructions in the mainline program from where the call to Macro is being made. This is known as Macro expansion. Due to this, the program code (which uses Macros) takes more memory space than the code which uses procedures for implementing the same task using the same set of instructions.

**DOS Interrupts**

Interrupt is the method of creating a temporary halt during program execution and allows peripheral devices to access the microprocessor. The microprocessor responds to that interrupt with an ISR (Interrupt Service Routine), which is a short program to instruct the microprocessor on how to handle the interrupt. The following image shows the types of interrupts we have in a 8086 microprocessor −

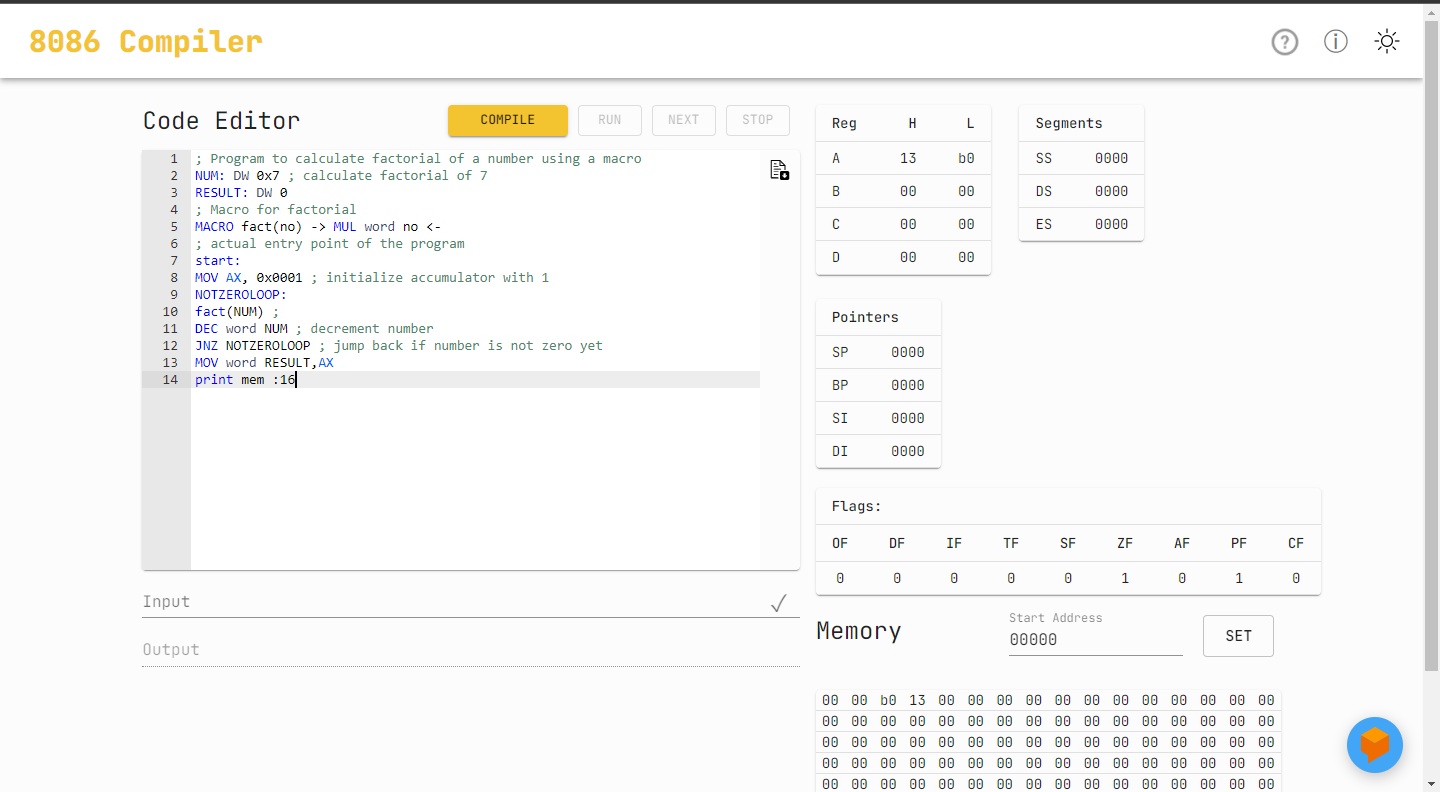
DOS Interrupt is a Software Interrupt. INT 21H is a DOS interrupt. It is one of the most commonly used interrupts while writing code in 8086 assembly language. To use the DOS interrupt 21H load AH with the desired sub-function. Load other required parameters in other registers, and make a call to INT 21H.

| AH | Description | AH | Description |
| --- | --- | --- | --- |
| 01 | Read character from STDIN | 02 | Write character to STDOUT |
| 05 | Write character to printer | 06 | Console Input/Output |
| 07 | Direct char read (STDIN), no echo | 08 | Char read from STDIN, no echo |
| 09 | Write string to STDOUT | 0A | Buffered input |
| 0B | Get STDIN status | 0C | Flush buffer for STDIN |
| 0D | Disk reset | 0E | Select default drive |
| 19 | Get current default drive | 25 | Set interrupt vector |
| 2A | Get system date | 2B | Set system date |
| 2C | Get system time | 2D | Set system time |
| 2E | Set verify flag | 30 | Get DOS version |
| 35 | Get Interrupt vector |  |  |
| 36 | Get free disk space | 39 | Create subdirectory |
| 3A | Remove subdirectory | 3B | Set working directory |
| 3C | Create file | 3D | Open file |
| 3E | Close file | 3F | Read file |
| 40 | Write file | 41 | Delete file |
| 42 | Seek file | 43 | Get/Set file attributes |
| 47 | Get current directory | 4C | Exit program |
| 4D | Get return code | 54 | Get verify flag |
| 56 | Rename file | 57 | Get/Set file date |

**CODE:**

**1. Factorial using Macros.**

| ; Program to calculate factorial of a number using a macro NUM: DW 0x7 ; calculate factorial of 7 RESULT: DW 0  ; Macro for factorial MACRO fact(no) -> MUL word no <- ; actual entry point of the program start: MOV AX, 0x0001 ; initialize accumulator with 1 NOTZEROLOOP: fact(NUM) ;  DEC word NUM ; decrement number JNZ NOTZEROLOOP ; jump back if number is not zero yet MOV word RESULT,AX print mem :16 |
| --- |



**2. DOS Interrupt**

| DATA SEGMENT NUM1 DB ? NUM2 DB ? RESULT DB ? MSG1 DB 10,13,"ENTER FIRST NUMBER TO ADD : $" MSG2 DB 10,13,"ENTER SECOND NUMBER TO ADD : $" MSG3 DB 10,13,"RESULT OF ADDITION IS : $" ENDS CODE SEGMENT ASSUME DS:DATA, CS:CODE START: MOV AX,DATA MOV DS,AX LEA DX,MSG1 MOV AH,9 INT 21H MOV AH,1 INT 21H SUB AL,30H MOV NUM1,AL LEA DX,MSG2 MOV AH,9 INT 21H MOV AH,1 INT 21H SUB AL,30H MOV NUM2,AL ADD AL,NUM1 MOV RESULT,AL MOV AH,0 AAA ADD AH,30H ADD AL,30H MOV BX,AX LEA DX,MSG3 MOV AH,9 INT 21H MOV AH,2 MOV DL,BH INT 21H MOV AH,2 MOV DL,BL INT 21H MOV AH,4CH INT 21H ENDS END START |
| --- |

