

**Artificial Intelligence and Data Science Department.**

MP / Even Sem 2021-22 / Experiment 5.

YASH SARANG.

47 / D6AD.

EXPERIMENT - 5.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**AIM:** Assembly programming using macro.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**THEORY:**

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.

For example, a very common need for programs is to write a string of characters on the screen. For displaying a string of characters, you need the following sequence of instructions −

mov edx,len ;message length

mov ecx,msg ;message to write

mov ebx,1 ;file descriptor (stdout)

mov eax,4 ;system call number (sys\_write)

int 0x80 ;call kernel

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Program:**

; A macro with two parameters

; Implements the write system call

%macro write\_string 2

mov eax, 4

mov ebx, 1

mov ecx, %1

mov edx, %2

int 80h

%endmacro

section .text

global \_start ;must be declared for using gcc

\_start: ;tell linker entry point

write\_string msg1, len1

write\_string msg2, len2

write\_string msg3, len3

mov eax,1 ;system call number (sys\_exit)

int 0x80 ;call kernel

section .data

msg1 db 'Hello, programmers!',0xA,0xD

len1 equ $ - msg1

msg2 db 'Welcome to the world of,', 0xA,0xD

len2 equ $- msg2

msg3 db 'Linux assembly programming! '

len3 equ $- msg3

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Output:**

When the above code is compiled and executed,   
it produces the following result −

Hello, programmers!

Welcome to the world of,

Linux assembly programming!

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**