# Name: Shrinivas Hatyalikar

# Div : CS-B

# Roll No: 24

**Title:** Execution of sample programs with procedure and Macro.

**Aim:-** Write 64 bit ALP for displaying the output messages using the macros and procedures

**Apparatus:**

* Core 2 duo/i3/i5/i7 - 64bit processor
* OS – ubuntu 32bit/64bit OS
* Assembler used –nasm (the netwide assembler)
* Editor Used – gedit

**Theory:** A procedure is a set of instructions that performs a specific task and can be called by the programmer when needed. The programmer uses a "call" function followed by the procedure name to execute the instructions. Using procedures makes code maintenance easier and improves efficiency.

Macros are similar to procedures, but they allow custom instructions to be created. When a macro is invoked by its name and parameters, the macro code is substituted in place of the name instead of passing control to the macro. However, using macros frequently can increase program space complexity and consume more memory. The syntax for defining a macro includes specifying the name and the number of parameters, and then writing the macro code followed by an "endmacro" statement.

**Code Using Macros:**

%macro print\_string 2 ; Define a macro that takes two arguments: a string and its length

mov rax, 1 ; Move the system call number for write into the rax register

mov rdi, 1 ; Move the file descriptor for stdout into the rdi register

mov rsi, %1 ; Move the address of the string into the rsi register

mov rdx, %2 ; Move the length of the string into the rdx register

syscall ; Call the kernel to write the string to stdout

%endmacro

section .data ; Declare the string to be displayed

name db "Name: Shrinivas Hatyalikar", 10

name\_len equ $-name ; Calculate the length of the string

rollno db "Roll No: 24", 10

rollno\_len equ $-rollno

prn db "PRN: 12110883", 10

prn\_len equ $-prn

subj db "Subject: COA", 10

subj\_len equ $-subj

section .text ; Start of code segment

global \_start

\_start:

print\_string name, name\_len ; Call the macro to display the string

print\_string rollno, rollno\_len

print\_string prn, prn\_len

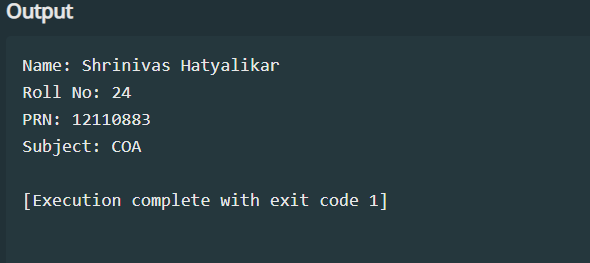
print\_string subj, subj\_len

mov rax, 60 ; Move the system call number for exit into the rax register

xor rdi, rdi ; Set the exit status to 0

syscall ; Call the kernel to exit

**Output:**



**Code Using Procedures:**

section .data ; Declare the string to be displayed

name db "Name: Shrinivas Hatyalikar", 10

name\_len equ $-name ; Calculate the length of the string

rollno db "Roll No: 24", 10

rollno\_len equ $-rollno

prn db "PRN: 12110883", 10

prn\_len equ $-prn

subj db "Subject: COA", 10

subj\_len equ $-subj

section .bss

section .text

global \_start

\_start:

mov rsi,name

mov rdx,name\_len

call print\_string

mov rsi,rollno

mov rdx,rollno\_len

call print\_string

mov rsi,prn

mov rdx,prn\_len

call print\_string

mov rsi,subj

mov rdx,subj\_len

call print\_string

mov rax,60

syscall

print\_string:

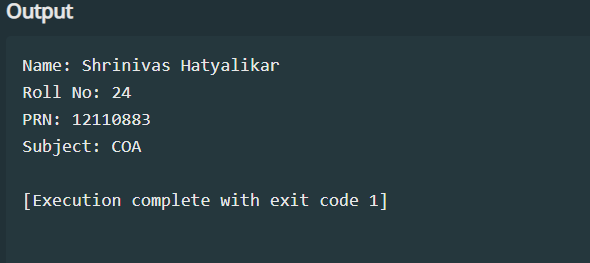
mov rax,01

mov rdi,01

syscall

ret

**Output:**



**Conclusion:** In programming, both procedures and macros have their own distinct purposes. Procedures are typically used to organize larger chunks of code and can be helpful for making changes to code that is frequently updated. However, using procedures too often can make the program slower. Macros, on the other hand, can simplify commonly used code sequences and make the program run more efficiently. However, using too many macros can increase the amount of space the program takes up. When deciding between procedures and macros, it's important to consider the size of the code and how often changes are made. Ultimately, the choice between the two will depend on the specific needs of the programming project.