ASSIGNMENT NO. 1

# Name: Akshat Chandrapatle

# Div.: CSA

# Roll No: 45

# Subject: COA

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

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

**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 (Macros):**

%macro display 2

mov rax, 1

mov rdi, 1

mov rsi, %1

mov rdx, %2

syscall

%endmacro

%macro exit 0

mov rax, 60

mov rdi, 0

syscall

%endmacro

section .data

msg db "Name: Akshat",10

len equ $-msg

msg1 db "Roll no: 45", 10

len1 equ $-msg1

msg2 db "Class: SY-CSA", 10

len2 equ $-msg2

msg3 db "PNR No: 12111449", 10

len3 equ $-msg3

msg4 db "Subject : COA", 10

len4 equ $-msg4

section .bss

section .text

global \_start

\_start:

display msg,len

display msg1, len1

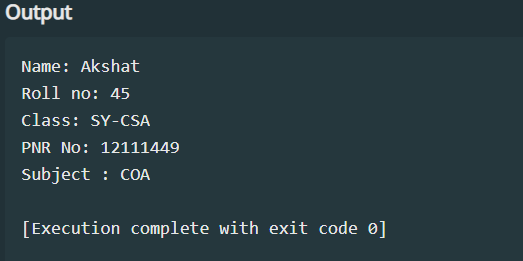
display msg2, len2

display msg3, len3

display msg4, len4

exit

**Output:**

****

**Code (Procedures):**

section .data

msg db "Name: Akshat",10

len equ $-msg

msg1 db "Roll no: 45", 10

len1 equ $-msg1

msg2 db "Class: SY-CSA", 10

len2 equ $-msg2

msg3 db "PNR No: 12111449", 10

len3 equ $-msg3

msg4 db "Subject : COA", 10

len4 equ $-msg4

section .bss

section .text

global \_start

\_start:

mov rsi,msg

mov rdx,len

call print

mov rsi,msg1

mov rdx,len1

call print

mov rsi,msg2

mov rdx,len2

call print

mov rsi,msg3

mov rdx,len3

call print

mov rsi,msg4

mov rdx,len4

call print

mov rax,60

syscall

print:

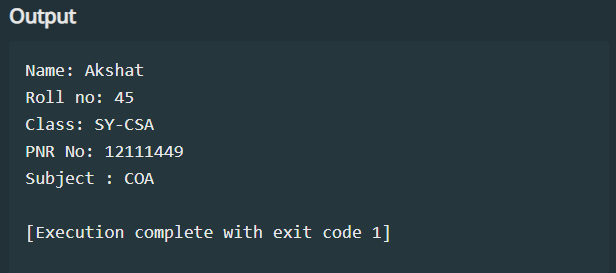
mov rax,01

mov rdi,01

syscall

ret

**Output-**



**Instructions:**

MOV: This instruction is used to move data from one location to another. Syntax – mov

destination, source

add: The add instruction adds together its two operands, storing the result in its first operand.

End: The END directive instructs the assembler to stop processing this source file

Functions:

09h: Displays the string until “$” is reached.

int 21h: Reads a character from the standard input device without copying it to the display. If no character is ready it waits until one is available.

.data: This Command is used only when we want to store in Data Segment, basically, it is the memory access of the Data Segment. Whatever we want to print must be written

10, 13: They work as Escape Sequence Character

$: It states the end of a Statement

**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.