section .data

msg1 db 10,"Roots are Complex",10

msglen1 equ $-msg1

msg2 db 10,"Root1: "

msglen2 equ $-msg2

msg3 db 10,"Root2: "

msglen3 equ $-msg3

a dd 5.00

b dd 3.00

c dd 3.00

four dd 4.00

two dd 2.00

hdec dq 100

dpoint db "."

section .bss

root1 resd 1

root2 resd 1

resbuff rest 1

temp resb 2

disc resd 1

char\_ans resb 2

%macro print 2

mov rax,1

mov rdi,1

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro read 2

mov rax,0

mov rdi,0

mov rsi,%1

mov rdx,%2

syscall

%endmacro

%macro exit 0

mov rax,60

xor rdi,rdi

syscall

%endmacro

section .text

global \_start

\_start:

finit

fld dword[b]

fmul dword[b]

fld dword[a]

fmul dword[c]

fmul dword[four]

fsub ftst ; compares ST0 and 0

fstsw ax ; Stores the coprocessor status word ;into either a word in memory or the AX register

sahf ; Stores the AH register into the FLAGS register. Loads the SF, ZF, AF, PF,

jb no\_real\_solutions ; if disc < 0, no real solutions

fsqrt ; stack: sqrt(b\*b - 4\*a\*c)

fst dword[disc] ; store disc= sqrt(b\*b - 4\*a\*c)

fsub dword[b] ; stack: disc-b

fdiv dword[a] ; stack: disc-b/2\*a or (-b+disc)/2a

fdiv dword[two]

print msg2,msglen2

call display\_result

fldz ;stack:0

fsub dword[disc] ;stack:-disc

fsub dword[b] ; stack: -disc - b

fdiv dword[a] ; stack: (-b - disc)/(2\*a)

fdiv dword[two]

print msg3,msglen3

call display\_result

jmp End

no\_real\_solutions:

print msg1,msglen1

End:

exit

display\_8:

mov rsi,char\_ans+1

mov rcx,2 ; number of digits

cnt: mov rdx,0

mov rbx,16

div rbx

cmp dl, 09h ; check for remainder in RDX

jbe add30

add dl, 07h

add30:

add dl,30h ; calculate ASCII code

mov [rsi],dl ; store it in buffer

dec rsi ; point to one byte back

dec rcx ; decrement count

jnz cnt

print char\_ans,2 ; display result on screen

ret

display\_result:

fimul dword[hdec]

fbstp tword[resbuff]

xor rcx,rcx

mov rcx,09H

mov rsi,resbuff+9

nextdigit:

push rcx

push rsi

xor rax,rax

mov al,[rsi]

call display\_8

pop rsi

dec rsi

pop rcx

loop nextdigit

print dpoint,1

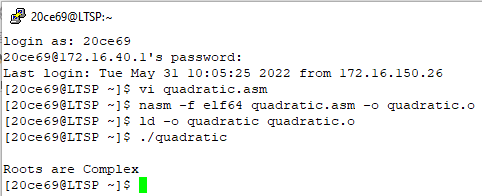
xor rax,rax

mov al,[resbuff]

call display\_8

ret

OUTPUT



2.mean variance

section .data

nline db 10,10

nline\_len: equ $-nline

msg db 10,"Program for Mean, Standard Deviation, & Variance"

db 10,"---------------------------------------------",10

msg\_len: equ $-msg

mmsg db 10,"CALCULATED MEAN : "

mmsg\_len equ $-mmsg

sdmsg db 10,"CALCULATED STANDARD DEVIATION : "

sdmsg\_len equ $-sdmsg

vmsg db 10,"CALCULATED VARIANCE : "

vmsg\_len equ $-vmsg

array dd 102.56, 198.21, 100.67, 230.78, 67.93

count dw 05

dpoint db '.'

hdec dq 100

section .bss

char\_ans resB 2

resbuff resT 1

mean resD 1

variance resD 1

;macros as per 64-bit convensions

%macro print 2

mov rax,1 ; Function 1 - write

mov rdi,1 ; To stdout

mov rsi,%1 ; String address

mov rdx,%2 ; String size

syscall ; invoke operating system to WRITE

%endmacro

%macro read 2

mov rax,0 ; Function 0 - Read

mov rdi,0 ; from stdin

mov rsi,%1 ; buffer address

mov rdx,%2 ; buffer size

syscall ; invoke operating system to READ

%endmacro

%macro exit 0

print nline, nline\_len

mov rax, 60 ; system call 60 is exit

xor rdi, rdi ; we want return code 0

syscall ; invoke operating system to exit

%endmacro

section .text

global \_start

\_start:

print msg, msg\_len

finit ; initialize coprocessor

fldz ; loads zero on top of stack st(0)=0

mov rbx,array

mov rsi,00 ; index of array initalized to 0

xor rcx,rcx

mov cx,[count] ; load count in cx reg

back: fadd dword[RBX+RSI\*4] ; st(0)+[array+(index\*4)]=st(0)

Double word = 4 bytes

inc rsi ; increment array index

loop back ; repeat addition untill all elements are added

fidiv word[count] ; st(0)=sum of array elements / count = mean

fst dword[mean] ; store the st(0) in mean

print mmsg,mmsg\_len

call display\_result

mov rbx,array

mov rsi,00 ; index of array initalized to 0

xor rcx,rcx

mov cx,[count] ; load count in cx reg

fldz ; loads zero on top of stack st(0)=0

back1:fldz

FLD DWORD[RBX+RSI\*4] ;st(0)=array[rsi]

FSUB DWORD[mean] ;st(0)=st(0)-mean

FST ST1

FMUL ;st(0)=st(0)\*st(1)

FADD ;add squared value to st(1) i.e. st(0)=st(0)+st(1)

INC RSI

LOOP back1

FIDIV word[count] ;divide result by count to get variance

FST dWORD[variance]

FSQRT ;st(0)=sqrt(st(0))= value of standard deviation

print sdmsg,sdmsg\_len

CALL display\_result

FLD dWORD[variance]

print vmsg,vmsg\_len

CALL display\_result

exit

display\_8:

mov rsi,char\_ans+1

mov rcx,2 ; number of digits

cnt: mov rdx,0

mov rbx,16

div rbx

cmp dl, 09h ; check for remainder in RDX

jbe add30

add dl, 07h

add30:

add dl,30h ; calculate ASCII code

mov [rsi],dl ; store it in buffer

dec rsi ; point to one byte back

dec rcx ; decrement count

jnz cnt

print char\_ans,2 ; display result on screen

ret

display\_result:

fimul dword[hdec]

fbstp tword[resbuff]

xor rcx,rcx

mov rcx,09H

mov rsi,resbuff+9

nextdigit:

push rcx

push rsi

xor rax,rax

mov al,[rsi]

call display\_8

pop rsi

dec rsi

pop rcx

loop nextdigit

print dpoint,1

xor rax,rax

mov al,[resbuff]

call display\_8

ret

OUTPUT

