## Problem Statement: Stack Operations in Assembly and C/C++Language

Description: This assembly language program implements a simple stack that can push and pop values. The program also includes a function to display the current top value of the stack.

**Complete the missing code and comments on the program. Transfer the assembly program using C++.**

**Assembly program:**

```
        .data                               ; Section for initialized data
stack:                                     ; Define a stack array
        .space    16                       ; Reserve 16 bytes for stack (4 elements of 32 bits
each)
top:                                          ; Define a variable to keep track of the stack
pointer
        .word      0                          ; Initialize stack pointer to 0 (indicating empty
stack)

        .text                              ; Section for code
        .global _start                ; Entry point of the program

_start:
        LDR R0, =stack                   ; Load address of stack into R0
        LDR R1, =top                       ; Load address of top variable into R1

        ; Push operation
        MOV R2, #88                         ; Value to push onto stack
        BL push                              ; Call push function

        MOV R2, #99                         ; Another value to push
        BL push                              ; Call push function

        ; Display top of stack
        BL display_top                   ; Call function to display the top value

        ; Pop operation
        BL pop                                ; Call pop function

        BL display_top                   ; Call function to display the new top value

        ; Exit program
        BX lr                                 ; Return from the program

; Function to push value onto the stack
push:
LDR R3, [R1]                         ; Load current top value from memory into R3
CMP R3, #4                           ; Check if stack is full (4 elements)
```

```
BEQ stack_full                  ; If full, branch to stack_full


STR R2, [R0, R3, LSL, #2]       ; Store value in stack at R3 * 4 (byte offset)
ADD R3, R3, #1                  ; Increment stack pointer
STR R3, [R1]                        ; Update top value in memory
BX lr                           ; Return from the function

stack_full:
    ; Handle stack full case (e.g., display error)
    MOV R0, #1                    ; Load error code
    BX lr                       ; Return from the function

; Function to pop value from the stack
pop:
    LDR R3, [R1]                  ; Load current top value in memory into R3
    CMP R3, #0                    ; Check if stack is empty
    BEQ stack_empty               ; If empty, branch to stack_empty

    SUB R3, R3, #1                ; Decrement pointer
    LDR R2, [R0, R3, LSL #2]      ; Load top value in stack (R3 * 4 bytes) into R2
    STR R3, [R1]                  ; Update top value in memory
    BX lr                        ; Return from function

stack_empty:
    ; Handle stack empty case (e.g., display error)
    MOV R0, #2                    ; Load error code
    BX lr                       ; Return from the function

; Function to display the top value of the stack
display_top:
    LDR R3, [R1]                    ; Load current top value from memory into R3
    CMP R3, #0                      ; Check if stack is empty
    SEQ stack_empty_display         ; If empty, branch to stack_empty_display

    LDR R2, [R0, R3, LSL #2]      ; Load top value from stack (R3 * 4 byte offset)
    BX lr                         ; Return from function

stack_empty_display:
    ; Handle stack empty case for display (e.g., display error)
    MOV R0, #3                    ; Load error code
    BX lr                       ; Return from the function
```