```c
#include <stdio.h>
#include <stdlib.h>

#define StackLimit 50

typedef int StackElementType;

typedef struct  {
    int Top;
    StackElementType Element[StackLimit];
} StackType;

typedef enum {
    FALSE, TRUE
} boolean;

void CreateStack(StackType *Stack);
void Push(StackType *Stack, StackElementType Item);
void Pop(StackType *Stack, StackElementType *Item);
boolean EmptyStack(StackType Stack);
boolean FullStack(StackType Stack);

int main()
{
    StackType myStack;
    StackElementType M,item,currentMemory;
    CreateStack(&myStack);

    /*??????? ????????? ??????(????? ???????).*/
    printf("Please enter maximum memory address:");
    scanf("%d",&M);

    /*???????? ??????????? ?????? ????? ?? ????? ? ???? 0(??????? ???????).*/
    do{
        printf("Please enter the next relative memory address:");
        scanf("%d",&item);

        Push(&myStack,item);
    }while(item != 0);

    /*"????????" ??? ????????? 0 ??? ??? ?????? ?? ????? ??????????????? ??
    ???? ??????????? ??? ??? ??????? ?? ????????.*/
    Pop(&myStack,&myStack.Element[myStack.Top]);

    /*???????? ????????? ?????? (????? ???????).*/
    printf("Please enter the current memory address:");
    scanf("%d",&currentMemory);

    /*???????? ??????? ???-??? ????? ?? ?????????? ???? ? ?????
    ? ????????? ?????? ?? ????????? ?? ????????? ????.*/
    while(myStack.Top != -1)
    {
        currentMemory+=myStack.Element[myStack.Top];
        Pop(&myStack,&myStack.Element[myStack.Top]);

        /*(??????? ???????)*/
        if((currentMemory >= 0) && (currentMemory <= M))
            printf("Executing instruction: %d\n",currentMemory);
        /*(?????? ???????)*/
        else
        {
            printf("Access Violation Exception at address:%d\n",currentMemory);
            break;
        }
    }

    return 0;
}

void CreateStack(StackType *Stack)
{
    Stack -> Top = -1;
   // (*Stack).Top = -1;
}

boolean EmptyStack(StackType Stack)
{
    return (Stack.Top == -1);
}

boolean FullStack(StackType Stack)
{
    return (Stack.Top == (StackLimit - 1));
}

void Push(StackType *Stack, StackElementType Item)
{
    if (!FullStack(*Stack)) {
        Stack -> Top++;
        Stack -> Element[Stack -> Top] = Item;
    } else
        printf("Full Stack...");
}

void Pop(StackType *Stack, StackElementType *Item)
{
    if (!EmptyStack(*Stack)) {
        *Item = Stack -> Element[Stack -> Top];
        Stack -> Top--;
    } else
        printf("Empty Stack...");
}
```