main:

sub sp, sp, #20 //Here, 20 Bytes of space are reserved on the stack, so that subsequent stack operations do not modify the local stack frame

mov r0, #0 //The mov instruction copies the data item referred to by its second operand

str r0, [sp, #16] //store a register value into memory

mov r1, #97 // copy It is an input operand that reserves a register for the variable assigns it to the placeholder

str r1, [sp, #12] // is used as a pointer to the active stack

mov r1, #65 // copy It is an input operand that reserves a register for the variable assigns it to the placeholder

strb r1, [sp, #11] //calculates an address from a base register value and an offset register value, and stores a byte from a register to memory.

mov r1, #98 copy It is an input operand that reserves a register for the variable assigns it to the placeholder

str r1, [sp, #4] is used as a pointer to the active stack

mov r1, #66 copy It is an input operand that reserves a register for the variable assigns it to the placeholder

strb r1, [sp, #3] calculates an address from a base register value and an offset register value, and stores a byte from a register to memory.

mov r1, #69 copy It is an input operand that reserves a register for the variable assigns it to the placeholder

strb r1, [sp, #2] calculates an address from a base register value and an offset register value, and stores a byte from a register to memory

ldr r1, [sp, #12] //to generate literal constants when an immediate value cannot be moved into a register because it is out of range of the MOV and MVN instructions

strb r1, [sp, #11] //calculates an address from a base register value and an offset register value, and stores a byte from a register to memory

ldr r1, [sp, #4] //to generate literal constants when an immediate value cannot be moved into a register because it is out of range of the MOV and MVN instructions

strb r1, [sp, #3] //calculates an address from a base register value and an offset register value, and stores a byte from a register to memory

mov r1, #81 //The mov instruction copies the data item referred to by its second operand

strb r1, [sp, #2] //calculates an address from a base register value and an offset register value, and stores a byte from a register to memory

ldr r1, [sp, #12] //to generate literal constants when an immediate value cannot be moved into a register because it is out of range of the MOV and MVN instructions

add r1, r1, #3 The add instruction adds together its two operands, storing the result in its first operand. Note, whereas both operands may be registers, at most one operand may be a memory location

str r1, [sp, #12] is used as a pointer to the active stack

ldr r1, [sp, #4] //to generate literal constants when an immediate value cannot be moved into a register because it is out of range of the MOV and MVN instructions

add r1, r1, #3 //The add instruction adds together its two operands, storing the result in its first operand. Note, whereas both operands may be registers, at most one operand may be a memory location

str r1, [sp, #4] //is used as a pointer to the active stack

add sp, sp, #20 //dd instruction adds together its two operands

bx lr //return from function