```
#*****************************************************************************
# File: es1.s
#       Contains the Assembly translation for es1.cpp.
#
# Author: Rambod Rahmani <rambodrahmani@autistici.org>
#         Created on 14/09/2019.
#*****************************************************************************

#-----------------------------------------------------------------------------
.TEXT
.GLOBAL _ZN2clC1Ec3st1                                  # cl::cl(char c, st1 s2)
#-----------------------------------------------------------------------------
# activation frame:
# -----------------
# i             -17
# s2            -13
# c             -9
# &this         -8
# %rbp          0
#-----------------------------------------------------------------------------
_ZN2clC1Ec3st1:
# set stack locations labels:
    .set this, -8
    .set c,    -9
    .set s2,   -13
    .set i,    -17

# prologue: activation frame
    pushq %rbp
    movq  %rsp, %rbp
    subq  $24, %rsp                 # reserve stack space for actual arguments

# copy actual arguments to the stack:
    movq %rdi, this(%rbp)
    movb %sil, c(%rbp)
    movl %edx, s2(%rbp)

# for loop initialization:
    movl $0, i(%rbp)                # i = 0

for:
    cmpl $4, i(%rbp)                # check if i < 4
    jge  finefor                    # end for loop (i >= 4)

# for loop body:
    movslq i(%rbp), %rcx            # i -> %rcx
    movq   this(%rbp), %rdi         # &this -> %rdi
    movb   c(%rbp), %al             # c -> %al
    movb   %al, 32(%rdi, %rcx, 1)   # s.vc[i] = c
    leaq   s2(%rbp), %rsi           # &s2 -> %rsi
    movsbq (%rsi, %rcx, 1), %rbx    # s2.vc[i] -> %rbx
    movsbq %al, %rax                # %al -> %rax
    subq   %rax, %rbx               # s2.vc[i] - c -> %rbx
    movq   %rbx, (%rdi, %rcx, 8)    # v[i] = s2.vc[i] - c;

    incl i(%rbp)                    # i++
    jmp  for                        # loop again

finefor:

    movq  this(%rbp), %rax          # return initialized object address
    leave                           # movq %rbp, %rsp; popq %rbp
    ret

#-----------------------------------------------------------------------------
.GLOBAL _ZN2cl5elab1ER3st1                              # void cl::elab1(st1& s1)
#-----------------------------------------------------------------------------
# activation frame:
# -----------------
# i             -60
```

```
# cla.v[0]      -56
# cla.v[1]      -48
# cla.v[2]      -40
# cla.v[3]      -32
# cla.s         -24
# &s1           -16
# &this         -8
# %rbp          0
#-------------------------------------------------------------------------
_ZN2cl5elab1ER3st1:
# set stack location labels:
    .set this, -8
    .set s1,   -16
    .set cla,  -56
    .set i,    -60

# prologue: activation frame
    pushq %rbp
    movq  %rsp, %rbp
    subq  $64, %rsp                 # reserve stack space for actual arguments

# copy actual arguments to the stack:
    movq %rdi, this(%rbp)
    movq %rsi, s1(%rbp)

# cl cla('x', s1);
    leaq cla(%rbp), %rdi
    movb $'x', %sil
    movq s1(%rbp), %rdx
    movl (%rdx), %edx
    call _ZN2clC1Ec3st1

# for loop initialization:
    movl $0, i(%rbp)                # i = 0

for1:
    cmpl $4, i(%rbp)                # check if i < 4
    jge  finefor1                   # end for loop (i >= 4)

# for loop body:
    movslq i(%rbp), %rcx            # i -> %rcx
    movq   s1(%rbp), %rsi           # &s1 -> %rsi
    movq   this(%rbp), %rdi         # &this -> %rdi
    movb   32(%rdi, %rcx, 1), %bl   # s.vc[i] -> %bl
    movb   (%rsi, %rcx, 1), %al     # s1.vc[i] -> %al
    cmpb   %al, %bl                 # compare s.vc[i] and s1.vc[i]
    jg     fineif                   # exit if (s.vc[i] > s1.vc[i])
    leaq   cla(%rbp), %rsi          # &cla -> %rsi
    movq   this(%rbp), %rdi         # &this -> %rdi
    movb   32(%rsi, %rcx, 1), %al   # cla.s.vc[i] -> %al
    movb   %al, 32(%rdi, %rcx, 1)   # s.vc[i] = cla.s.vc[i]
    movq   (%rsi, %rcx, 8), %rax    # cla.v[i] -> %rax
    addq   %rcx, %rax               # cla.v[i] + i -> %rax
    movq   %rax, (%rdi, %rcx, 8)    # v[i] = cla.v[i] + i;

fineif:

    incl i(%rbp)                    # i++
    jmp  for1                       # loop again

finefor1:

    leave                           # movq %rbp, %rsp; popq %rbp
    ret
#*************************************************************************
```