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

#-----------------------------------------------------------------------------
.TEXT
.GLOBAL _ZN2clC1EPc                                          # cl::cl(char v[])
#-----------------------------------------------------------------------------
# activation frame:
# -----------------
# i             -20
# &v            -16
# &this         -8
# %rbp          0
#-----------------------------------------------------------------------------
_ZN2clC1EPc:
# set stack location labels:
    .set this, -8
    .set v,    -16
    .set i,    -20

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

# copy actual arguments to the stack
    movq %rdi, this(%rbp)
    movq %rsi, v(%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
    movq   v(%rbp), %rsi            # &v -> %rsi
    movsbq (%rsi, %rcx, 1), %rax    # v[i] -> %rax
    movb   %al, (%rdi, %rcx, 1)     # s.vv1[i] = v[i];
    movq   %rax, 8(%rdi, %rcx, 8)   # s.vv2[i] = v[i]

    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 _ZN2cl5elab1EiR2st                      # void cl:: elab1(int d, st& ss)
#-----------------------------------------------------------------------------
# activation frame:
# -----------------
# i             -28
# &ss           -24
# d             -12
# &this         -8
# %rbp          0
#-----------------------------------------------------------------------------
_ZN2cl5elab1EiR2st:
```

```
# set stack locations labels:
    .set this, -8
    .set d,    -12
    .set ss,   -24
    .set i,    -28

# prologue: activation frame
    pushq %rbp
    movq  %rsp, %rbp
    subq  $32, %rsp                 # reserve stack space for actual arguments

# copy actual arguments to the stack:
    movq %rdi, this(%rbp)
    movl %esi, d(%rbp)
    movq %rdx, ss(%rbp)

# 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:
    movq   this(%rbp), %rdi         # &this -> %rdi
    movslq i(%rbp), %rcx            # i -> %rcx
    movslq d(%rbp), %rdx            # d -> %rdx
    movq   ss(%rbp), %rsi           # &ss -> %rsi

# if (d >= ss.vv2[i])
    movq   8(%rsi, %rcx, 8), %rax   # ss.vv2[i] -> %rax
    cmpq   %rax, %rdx               # compare d and ss.vv2[i]
    jl     fineif                   # exit if (d < ss.vv2[i])
    movb   (%rsi, %rcx, 1), %bl     # ss.vv1[i] -> %bl
    addb   %bl, (%rdi, %rcx, 1)     # s.vv1[i] += ss.vv1[i]

fineif:

    addq   %rcx, %rdx               # d + i -> %rdx
    movq   %rdx, 8(%rdi, %rcx, 8)   # s.vv2 = d + i;

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

finefor1:

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

#*****************************************************************************
```