```
#*****************************************************************************
# 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 record:
# ------------------
#  i            -20
#  &v           -16
#  this         -8
#  %rbp         0
#-----------------------------------------------------------------------------
_ZN2clC1EPc:
# set stack locations labels
    .set this, -8
    .set v,    -16
    .set i,    -20

# 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)
    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
    movq   this(%rbp), %rdi         # this -> %rdi
    movq   v(%rbp), %rsi            # &v -> %rsi
    movslq i(%rbp), %rcx            # i => %rcx
    movq   $3, %r8                  # 3 -> %r8
    subq   %rcx, %r8                # 3 - i -> %r8
    movb   (%rsi, %r8, 1), %al      # v[3 - i] -> %al
    movsbq %al, %rax                # v[3 - i] => %rax
    movq   %rax, (%rdi, %rcx, 8)    # s.vv2[i] = v[3 - i];
    movq   %rax, 32(%rdi, %rcx, 1)  # s.vv1[i] = v[3 - i];

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

finefor:

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

#-----------------------------------------------------------------------------
.GLOBAL _ZN2cl5elab1ER2sti
#-----------------------------------------------------------------------------
# activation record:
# ------------------
#  i            -24
#  d            -20
#  &ss          -16
#  this         -8
#  %rbp         0
```

```
#------------------------------------------------------------------------------
_ZN2cl5elab1ER2sti:
# set stack locations labels
    .set this, -8
    .set ss,   -16
    .set d,    -20
    .set i,    -24

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

# copy actual arguments to the stack
    movq %rdi, this(%rbp)
    movq %rsi, ss(%rbp)
    movl %edx, d(%rbp)

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

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

# for loop body
    movq   this(%rbp), %rdi         # &this -> %rdi
    movq   ss(%rbp), %rsi           # &ss -> 5rsi
    movslq d(%rbp), %rdx            # d => %rdx
    movslq i(%rbp), %rcx            # i => %rcx

# if (d >= ss.vv2[i])
    movq  (%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  32(%rsi, %rcx, 1), %bl    # ss.vv1[i] -> %bl
    addb  %bl, 32(%rdi, %rcx, 1)    # s.vv1[i] += ss.vv1[i];

fineif:

    subq %rcx, %rdx                 # d - i -> %rdx
    movq %rdx, (%rdi, %rcx, 8)      # s.vv2[i] = d - i;

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

finefor1:

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

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