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

#-----------------------------------------------------------------------------
.TEXT
.GLOBAL _ZN2clC1EcR3st2                                  # cl::cl(char c, st2& s2)
#-----------------------------------------------------------------------------
# activation frame:
# -----------------
#  i            -28
#  &s2          -24
#  c            -9
#  this         -8
#  %rbp         0
#-----------------------------------------------------------------------------
_ZN2clC1EcR3st2:
# set stack locations labels:
    .set this, -8
    .set c,    -9
    .set s2,   -24
    .set i,    -28

# prologue: activation frame
    pushq %rbp
    movq  %rsp, %rbp
    subq  $28, %rsp                 # reserve stack space for actual arguments

# copy actual arguments to the stack
    movq %rdi, this(%rbp)
    movb %sil, c(%rbp)
    movq %rdx, 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
    movq   this(%rbp), %rdi         # this -> %rdi
    movq   s2(%rbp), %rsi           # &s2 -> %rsi
    movslq i(%rbp), %rcx            # i => %rcx
    movb   c(%rbp), %al             # c -> %al
    addb   %cl, %al                 # c + i -> %al
    movb   %al, (%rdi, %rcx, 1)     # s.vc[i] = c + i;
    movsbl (%rdi, %rcx, 1), %ebx    # s.vc[i] -> %bl
    movl   (%rsi, %rcx, 4), %eax    # s2.vd[i] -> %eax
    addl   %ebx, %eax               # s2.vd[i] + s.vc[i] -> %eax
    movslq %eax, %rax               # %eax => %rax
    movq   %rax, 8(%rdi, %rcx, 8)   # v[i] = s2.vd[i] + s.vc[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 _ZN2cl5elab1E3st13st2                  # void cl::elab1(st1 s1, st2 s2)
#-----------------------------------------------------------------------------
# activation frame:
```

```
# -----------------
#  i            -76
#  cla.s        -72
#  cla.v[0]     -64
#  cla.v[1]     -56
#  cla.v[2]     -48
#  cla.v[3]     -40
#  s2 [MSB]     -32
#  s2 [LSB]     -24
#  s1           -12
#  this         -8
#  %rbp         0
#-------------------------------------------------------------------------------
_ZN2cl5elab1E3st13st2:
# set stack locations labels
    .set this, -8
    .set s1,   -16
    .set s2,   -32
    .set cla,  -72
    .set i,    -76

# prologue: activation frame
    pushq %rbp
    movq  %rsp, %rbp
    subq  $80, %rsp                 # reserve stack space for actual arguments

# copy actual arguments to the stack
    movq %rdi, this(%rbp)
    movl %esi, s1(%rbp)
    movq %rdx, s2(%rbp)
    movq %rcx, -24(%rbp)

# cl cla('a', s2);
    leaq cla(%rbp), %rdi
    movb $'a', %sil
    leaq s2(%rbp), %rdx
    call _ZN2clC1EcR3st2

# for loop 1 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 1 body
# if (s.vc[i] <= s1.vc[i])
    movq   this(%rbp), %rdi         # this -> %rdi
    movslq i(%rbp), %rcx            # i => %rcx
    leaq   s1(%rbp), %rsi           # &s1 -> %rsi
    movb   (%rsi, %rcx, 1), %al     # s1.vc[i] -> %al
    movb   (%rdi, %rcx, 1), %bl     # s.vc[i] -> %bl
    cmpb   %al, %bl                 # compare s.vc[i] and s1.vc[i]
    jg     fineif                   # exit if (s.vc[i] > s1.vc[i])

    # if body #
    leaq cla(%rbp), %r8
    movb (%r8, %rcx, 1), %al        # cla.s.vc[i] -> %al
    movb %al, (%rdi, %rcx, 1)       # s.vc[i] = cla.s.vc[i];

    movq 8(%r8, %rcx, 8), %rbx      # cla.v[i] -> %rbx
    leaq 8(%rdi), %r9               # &v -> %r9
    movq %rbx, (%r9, %rcx, 8)       # v[i] = cla.v[i];


fineif:

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

```
finefor1:

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

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