```
#*******************************************************************************
# File: es1.s
#       Contains the Assembly translation for es1.cpp.
#
# Author: Rambod Rahmani <rambodrahmani@autistici.org>
#         Created on 24/08/2019.
#*******************************************************************************

#-------------------------------------------------------------------------------
.TEXT
.GLOBAL _ZN2clC1EcR3st1                               # cl::cl(char c, st1 & s2)
#-------------------------------------------------------------------------------
# activation record:
# ------------------
#  &s2         -24
#  c           -16
#  i           -12
#  &this       -8
#  %rbp        0
#-------------------------------------------------------------------------------
_ZN2clC1EcR3st1:
# set stack locations labels
    .set  this, -8
    .set  i,    -12
    .set  c,    -16
    .set  s2,   -24

# 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, c(%rbp)
    movq %rdx, s2(%rbp)

# for loop 1 initialization
    movl $0, i(%rbp)                 # i = 0

for1:
    cmpl $8, i(%rbp)                 # check if i < 4
    jge  finefor1                    # end for loop (i >= 8)

# for loop 1 body
    movq   this(%rbp), %rdi          # this -> %rdi
    movslq i(%rbp), %rcx
    movb   c(%rbp), %al              # c -> %al
    addb   i(%rbp), %al              # c + i -> %al
    movb   %al, (%rdi, %rcx, 1)      # s.vc[i] = c + i

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

finefor1:

# for loop 2, initialization
    movl $0, i(%rbp)                 # i = 0

for2:
    cmpl $4, i(%rbp)                 # check if i < 4
    jge  finefor2                    # end for loop (i >= 4)

# for loop 2, body
    movq   this(%rbp), %rdi          # this -> %rdi
    movslq i(%rbp), %rcx             # i -> %rcx
    movq   s2(%rbp), %rsi            # &s2 -> %rsi

    movb   (%rsi, %rcx, 1), %al      # s.vc[i] -> %al
    subb   (%rdi, %rcx, 1), %al      # s2.vc[i] - s.vc[i] -> %al
    movsbl %al, %eax                  # %al -> %eax
    movl   %eax, 8(%rdi, %rcx, 4)     # v[i] = s2.vc[i] - s.vc[i];

    incl   i(%rbp)                    # i++
    jmp    for2                       # loop again

finefor2:
    movq   this(%rbp), %rax           # return initialized object address
    leave                             # movq %rbp, %rsp; popq %rbp
    ret

#--------------------------------------------------------------------------------
.GLOBAL _ZN2cl5elab1E3st1R3st2                 # void cl::elab1(st1 s1, st2 & s2)
#--------------------------------------------------------------------------------
# activation record:
# ------------------
#  i           -52
#  cla_s       -48
#  cla_v       -40
#  &s2         -24
#  s1 LSB      -16
#  s1 MSB      -12
#  &this       -8
#  %rbp        0
#--------------------------------------------------------------------------------
_ZN2cl5elab1E3st1R3st2:
# set stack locations labels
    .set this,  -8
    .set s1,    -16
    .set s2,    -24
    .set cla_v, -40
    .set cla_s, -48
    .set i,     -52

# prologue: activation frame
    pushq %rbp
    movq  %rsp, %rbp
    subq  $56, %rsp                    # reserve stack space for actual arguments

# copy actual arguments to the stack
    movq  %rdi, this(%rbp)
    movq  %rsi, s1(%rbp)
    movq  %rdx, s2(%rbp)

# prepare actual arguments to call constructor
    leaq  -48(%rbp), %rdi              # &cla
    movb  $'f', %sil                   # 'f'
    leaq  s1(%rbp), %rdx               # s1
    call  _ZN2clC1EcR3st1              # cl cla('f', s1);

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

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

# for loop, body
if1:
    movslq i(%rbp), %rcx
    movq   this(%rbp), %rsi
    leaq   s1(%rbp), %rdi

    movb   (%rdi, %rcx, 1), %bl        # s1.vc[i] -> %bl
    movb   (%rsi, %rcx, 1), %al        # s.vc[i] -> %al
    cmpb   %al, %bl                    # if (s.vc[i] > s1.vc[i])
    jl     fineif1                     # exit if

# if1 body
    leaq   cla_s(%rbp), %rdi           # &cla.s.vc[i] -> %rsi
    movb   (%rdi, %rcx, 1), %al       # cla.s.vc[i] -> %cl
    movb   %al, (%rsi, %rcx, 1)       # s.vc[i] = cla.s.vc[i];

fineif1:

#if2:
    leaq  cla_v(%rbp), %rsi
    movl  (%rsi, %rcx, 4), %eax
    movq  this(%rbp), %rdi
    movl  8(%rdi, %rcx, 4), %ebx      # this.v[i] -> %ebx
    cmpl  %ebx, %eax                  # if (v[i] > cla.v[i])
    jl    fineif2                     # exit if

# if2 body
    addl  i(%rbp), %eax               # cla.v[i] + i -> %eax
    movl  %eax, 8(%rdi, %rcx, 4)

fineif2:
    incl  i(%rbp)                     # i++
    jmp   for                         # loop again

finefor:

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