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

#--------------------------------------------------------------------------------
.TEXT
.GLOBAL _ZN2clC1Ec3st2                                  # cl::cl(char c, st2 s2)
#--------------------------------------------------------------------------------
# activation record:
# ------------------
#  i            -40
#  s2 [MSB]     -32
#  s2 [LSB]     -24
#  c            -9
#  &this        -8
#  %rbp         0
#--------------------------------------------------------------------------------
_ZN2clC1Ec3st2:
# set stack locations labels
    .set this, -8
    .set c,    -9
    .set s2,   -32
    .set i,    -40

# prologue: activation record
    pushq %rbp
    movq  %rsp, %rbp
    subq  $40, %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)
    movq %rcx, s2+8(%rbp)

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

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

# for loop body
    movq   this(%rbp), %rdi         # &this -> %rdi
    movslq i(%rbp), %rcx            # i => %rcx
    movb   c(%rbp), %al             # c -> %al
    movb   %al, (%rdi, %rcx, 1)     # s.vc[i] = c;
    movb   (%rdi, %rcx, 1), %bl     # s.vc[i] -> %bl
    movsbl %bl, %ebx                # %bl => %ebx
    leaq   s2(%rbp), %rsi           # &s2 -> %rsi
    movl   (%rsi, %rcx, 4), %eax    # s2.vd[i] -> %eax
    subl   %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

finefor:

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

#--------------------------------------------------------------------------------
.GLOBAL _ZN2cl5elab1E3st1R3st2                 # void cl::elab1(st1 s1, st2& s2)
#--------------------------------------------------------------------------------
```

```
# activation record:
# ------------------
#  i             -68
#  cla.s         -64
#  cla.v[0]      -56
#  cla.v[1]      -48
#  cla.v[2]      -40
#  cla.v[3]      -32
#  &s2           -24
#  s1            -16
#  &this         -8
#  %rbp          0
#------------------------------------------------------------------------------------
_ZN2cl5elab1E3st1R3st2:
# set stack locations labels
    .set this, -8
    .set s1,   -16
    .set s2,   -24
    .set cla,  -64
    .set i,    -68

# prologue: activation record
    pushq %rbp
    movq  %rsp, %rbp
    subq  $72, %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)

# cl cla('f', s2);
    leaq cla(%rbp), %rdi
    movb $'f', %sil
    movq s2(%rbp), %r8
    movq (%r8), %rdx
    movq 8(%r8), %rcx
    call _ZN2clC1Ec3st2

# 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
    leaq   s1(%rbp), %rsi           # &s1 -> %rsi

# if (s.vc[i] < s1.vc[i])
    movb (%rsi, %rcx, 1), %al       # s1.vc[i] -> %al
    movb (%rdi, %rcx, 1), %bl       # s.vc[i] -> %bl
    cmpb %bl, %al                   # compare s.vc[i] and s1.vc[i]
    jle  fineif1                    # exit if (s1.vc[i] <= s.vc[i])
    movb cla(%rbp, %rcx, 1), %al    # cla.s.vc[i]; -> %al
    movb %al, (%rdi, %rcx, 1)       # s.vc[i] = cla.s.vc[i];

fineif1:

# if (v[i] <= cla.v[i])
    leaq cla(%rbp), %rsi            # &cla -> %rsi
    movq 8(%rsi, %rcx, 8), %rax     # cla.v[i] -> %rax
    movq 8(%rdi, %rcx, 8), %rbx     # v[i] -> %rbx
    cmpq %rax, %rbx                 # compare v[i] and cla.v[i]
    jg   fineif2                    # exit if (v[i] > cla.v[i])
    addq %rax, 8(%rdi, %rcx, 8)     # v[i] += cla.v[i];

fineif2:
    incl i(%rbp)
    jmp  for1

finefor1:

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

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