```
#*****************************************************************************
# File: es1.s
#       Contains the Assembly translation for es1.cpp.
#
# Author: Rambod Rahmani <rambodrahmani@autistici.org>
#         Created on 10/09/2019.
#*****************************************************************************

#-----------------------------------------------------------------------------
.TEXT
.GLOBAL _ZN2clC1E3st1                                          # cl:cl(st1 ss)
#-----------------------------------------------------------------------------
# activation record:
# ------------------
#  i           -16
#  ss          -12
#  this        -8
#  %rbp        0
#-----------------------------------------------------------------------------
_ZN2clC1E3st1:
# set stack locations labels
    .set this, -8
    .set ss,   -12
    .set i,    -16

# prologue: activation frame
    pushq %rbp
    movq  %rsp, %rbp
    subq  $16, %rsp                  # reserve stack space for actual arguments

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

# for loop initialization
    movl $0, i(%rbp)                 # i = 0
    movq $0, %rax                    # clear out %rax

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

# for loop body
    movq   this(%rbp), %rsi
    movslq i(%rbp), %rcx
    leaq   ss(%rbp), %rdx              # &ss -> %rdx
    movb   (%rdx, %rcx, 1), %al        # ss.vi[i] -> %al
    movb   %al, (%rdi, %rcx, 1)        # v1[i] = ss.vi[i]
    movb   %al, 4(%rdi, %rcx, 1)       # v2[i] = ss.vi[i]
    addl   %eax, %eax                  # ss.vi[i] + ss.vi[i] -> %eax
    movq   %rax, 8(%rdi, %rcx, 8)      # v3[i] = ss.vi[i] + ss.vi[i]

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

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

#-----------------------------------------------------------------------------
.GLOBAL _ZN2clC1E3st1Pl                           # cl::cl(st1 s1, long ar2[])
#-----------------------------------------------------------------------------
# activation record:
# ------------------
#  i           -28
#  &ar2        -24
#  s1          -12
#  this        -8
#  %rbp        0
```

```
#-------------------------------------------------------------------------------
_ZN2clC1E3st1Pl:
# set stack locations labels
    .set this, -8
    .set s1,   -12
    .set ar2,  -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)
    movl %esi, s1(%rbp)
    movq %rdx, ar2(%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
    movslq i(%rbp), %rcx
    leaq   s1(%rbp), %rdx            # &s1 -> %rdx
    movb   (%rdx, %rcx, 1), %al      # s1.vi[i] -> %al
    movb   %al, (%rdi, %rcx, 1)      # v1[i] = s1.vi[i]
    movb   %al, 4(%rdi, %rcx, 1)     # v2[i] = s1.vi[i]
    movq   ar2(%rbp), %rsi           # &ar2 -> %rsi
    movq   (%rsi, %rcx, 8), %rbx     # ar2[i] -> %rbx
    movq   %rbx, 8(%rdi, %rcx, 8)    # v3[i] = ar2[i]

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

finefor1:

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

#-------------------------------------------------------------------------------
.GLOBAL _ZN2cl5elab1EPc3st2                  # cl cl::elab1(char ar1[], st2 s2)
#-------------------------------------------------------------------------------
# activation record:
# ------------------
#  i            -92
#  cla.v1/v2    -88
#  cla.v3[0]    -80
#  cla.v3[1]    -72
#  cla.v3[2]    -64
#  cla.v3[3]    -56
#  s1           -48
#  s2 [MSB]     -40
#  s2 [LSB]     -32
#  &ar1         -24
#  this         -16     <- this (cl object) address
#  indo         -8      <- leave returned cl object address here
#  %rbp         0
#-------------------------------------------------------------------------------
_ZN2cl5elab1EPc3st2:
# set stack locations labels
    .set indo,   -8
    .set this,   -16
    .set ar1,    -24
    .set s2,     -40
```

```
    .set s1,     -48
    .set cla_v3, -80
    .set cla_v2, -84
    .set cla_v1, -88
    .set i,      -92

# prologue: activation frame
    pushq %rbp
    movq  %rsp, %rbp
    subq  $96, %rsp                  # reserve space stack for actual arguments

# copy actual arguments to the stack
    movq %rdi, indo(%rbp)
    movq %rsi, this(%rbp)
    movq %rdx, ar1(%rbp)
    movq %rcx, s2(%rbp)
    movq %r8, -32(%rbp)

# for loop 1 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 1 body
    movslq i(%rbp), %rcx             # i --64ext--> %rcx
    movq   ar1(%rbp), %rdi           # &ar1 -> %rsi
    movb   (%rdi, %rcx, 1), %al      # ar1[i] -> %al
    leaq   s1(%rbp), %rsi            # &s1 -> %rax
    movb   %al, (%rsi, %rcx, 1)      # s1.vi[i] = ar1[i]

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

finefor2:

# prepare actual arguments to call constructor
    leaq  cla_v1(%rbp), %rdi         # leave &this in %rdi
    movl  s1(%rbp), %esi             # leave ss in %rsi
    call  _ZN2clC1E3st1              # cl cla(s1);

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

for3:
    cmpl $4, i(%rbp)
    jge  finefor3

# for loop 2 body
    movslq i(%rbp), %rcx             # i -> %rcx
    leaq   s2(%rbp), %rdx            # &s2 -> %rax
    movl   (%rdx, %rcx, 4), %ebx     # s2.vd[i] -> %ebx
    movslq %ebx, %rbx
    movq   %rbx, cla_v3(%rbp, %rcx, 8)

    incl  i(%rbp)                    # i++
    jmp   for3                       # loop again

finefor3:

# copy return object from stack to the address in indo
    leaq cla_v1(%rbp), %rsi          # rep movsq source address
    movq indo(%rbp), %rdi            # rep movsq destination address
    movabsq $5, %rcx                 # rep movsq repetitions
    rep movsq                        # rep movsq, [0]
    movq indo(%rbp), %rax            # return initialized object address

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

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

#############################################################################
# [0]
# Copies the quad word address by %rsi into the location addressed by %rdi. It
# will then increment both %rsi and %rdi and repeat. The number of repetitions
# is set using %rcx.
#############################################################################
```