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

#-----------------------------------------------------------------------------
.TEXT
.GLOBAL _ZN2clC1E3st1                                          # cl::cl(st1 ss)
#-----------------------------------------------------------------------------
# activation frame:
# -----------------
#  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                # reserver stack space for actual arguments

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

# foor 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
    leaq   ss(%rbp), %rsi          # &ss -> %rsi
    movslq i(%rbp), %rcx           # i => %rcx
    movb   (%rsi, %rcx, 1), %al    # ss.vi[i] -> %al
    movb   %al, (%rdi, %rcx, 1)    # v1[i] = ss.vi[i];
    movsbq %al, %rax               # %al => %rax
    movq   %rax, 24(%rdi, %rcx, 8) # v2[i] = ss.vi[i];
    movsbl %al, %eax               # %al => %eax
    sal    $2, %eax                # 4 * ss.vi[i] -> %eax
    movl   %eax, 4(%rdi, %rcx, 4)  # v3[i] = 4 * ss.vi[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 _ZN2clC1ER3st1Pi                          # cl::cl(st1& s1, int ar2[])
#-----------------------------------------------------------------------------
# activation record:
# ------------------
#  i            -28
#  &ar2         -24
#  &s1          -16
```

```
#  this         -8
#  %rbp         0
#-----------------------------------------------------------------------------
_ZN2clC1ER3st1Pi:
# set stack locations labels
    .set this, -8
    .set s1,   -16
    .set ar2,  -24
    .set i,    -28

# prologue: activation frame
    pushq %rbp
    movq  %rsp, %rbp
    subq  $32, %rsp                 # reserve stack space for actual arguments

# copy actual arguments to the stack
    movq %rdi, this(%rbp)
    movq %rsi, s1(%rbp)
    movq %rdx, ar2(%rbp)

# for loop intialization
    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
    movq   s1(%rbp), %rsi           # &s1 -> %rsi
    movq   ar2(%rbp), %rdx          # &ar2 -> %rdx
    movslq i(%rbp), %rcx            # i => %rcx
    movb   (%rsi, %rcx, 1), %al     # s1.vi[i] -> %al
    movb   %al, (%rdi, %rcx, 1)     # v1[i] = s1.vi[i];
    neg    %al                      # -s1.vi[i] -> %al
    movsbq %al, %rax                # %al => %rax
    movq   %rax, 24(%rdi, %rcx, 8)  # v2[i] = -s1.vi[i];
    movl   (%rdx, %rcx, 4), %eax    # ar2[i] -> %eax
    movl   %eax, 4(%rdi, %rcx, 4)   # v3[i] = ar2[i];

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

finefor1:

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

#-----------------------------------------------------------------------------
.GLOBAL _ZN2cl5elab1EPcRK3st2          # cl cl::elab1(char ar1[], const st2& s2)
#-----------------------------------------------------------------------------
# activation frame:
# -----------------
#  i            -100
#  cla.v1       -96
#  cla.v3[0-1]  -92
#  cla.v3[2-3]  -84
#  cla.v2[0]    -72
#  cla.v2[1]    -64
#  cla.v2[2]    -56
#  cla.v2[3]    -48
#  s1           -40
#  &s2          -32
#  &ar1         -24
#  this         -16
#  indo         -8
#  %rbp         0
#-----------------------------------------------------------------------------
_ZN2cl5elab1EPcRK3st2:
# set stack locations labels
```

```
    .set indo, -8
    .set this, -16
    .set ar1,  -24
    .set s2,   -32
    .set s1,   -40
    .set cla,  -96
    .set i,    -100

# prologue: activation frame
    pushq %rbp
    movq  %rsp, %rbp
    subq  $100, %rsp                # reserve stack space 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)

# 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
    movq   this(%rbp), %rdi         # &this -> %rdi
    movq   ar1(%rbp), %rsi          # &ar1 -> %rsi
    movq   s2(%rbp), %rdx           # &s2 -> %rdx
    movslq i(%rbp), %rcx            # i => %rcx
    movb   (%rsi, %rcx, 1), %al     # ar1[i] -> %al
    subb   %cl, %al                 # ar1[i] - i -> %al
    leaq   s1(%rbp), %r8            # &s1 -> %r8
    movb   %al, (%r8, %rcx, 1)      # s1.vi[i] = ar1[i] - i;

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

finefor2:

# cl cla(s1);
    leaq cla(%rbp), %rdi
    movl s1(%rbp), %esi
    call _ZN2clC1E3st1

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

for3:
    cmpl $4, i(%rbp)                # check if i < 4
    jge  finefor3                   # end for loop (i >= 4)

# for loop 2 body
    movq   this(%rbp), %rdi         # &this -> %rdi
    leaq   -92(%rbp), %rsi          # &cla.v3 -> %rsi
    movq   s2(%rbp), %rdx           # &s2 -> %rdx
    movslq i(%rbp), %rcx            # i => %rcx
    movq   (%rdx, %rcx, 8), %rax    # s2.vd[i] -> %rax
    movl   %eax, (%rsi, %rcx, 4)    # cla.v3[i] = s2.vd[i];

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

finefor3:

# copy return object from stack to the address in indo
    leaq cla(%rbp), %rsi            # rep movsq source address
    movq indo(%rbp), %rdi           # rep movsq destination address
    movabsq $7, %rcx                # rep movsq repetitions
```

```
    rep movsq                       # rep movsq, [0]
    movq indo(%rbp), %rax           # return initialized object address

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

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