```
#*****************************************************************************
# 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 record:
# ------------------
#  i            -36
#  ss [MSB]     -32
#  ss [LSB]     -16
#  &this        -8
#  %rbp         0
#-----------------------------------------------------------------------------
_ZN2clC1E3st1:
# set stack locations labels:
    .set this, -8
    .set ss,   -32
    .set i,    -36

# 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)
    movq %rsi, ss(%rbp)
    movq %rdx, ss+8(%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
    movslq i(%rbp), %rcx            # i -> %rcx
    leaq   ss(%rbp), %rsi           # &ss -> %rsi
    movslq (%rsi, %rcx, 4), %rax    # ss.vi[i] -> %rax
    movb   %al, (%rdi, %rcx, 1)     # v1[i] = ss.vi[i]
    sar    $1, %rax                 # ss.vi[i]/2 -> %rax
    movq   %rax, 8(%rdi, %rcx, 8)   # v2[i] = ss.vi[i]/2
    movslq (%rsi, %rcx, 4), %rax    # ss.vi[i] -> %rax
    sal    $1, %rax                 # 2*ss.vi[i] -> %rax
    movb   %al, 4(%rdi, %rcx, 1)    # v3[i] = 2*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 frame:
# -----------------
#  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

# copy actual arguments to the stack:
    movq %rdi, this(%rbp)
    movq %rsi, 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         # &this -> %rdi
    movslq i(%rbp), %rcx            # i -> %rcx
    movq   s1(%rbp), %rsi           # &s1 -> %rsi
    movslq (%rsi, %rcx, 4), %rax    # s1.vi[i] -> %rax
    movb   %al, (%rdi, %rcx, 1)     # v1[i] = s1.vi[i]
    sar    $2, %rax                 # s1.vi[i]/4 -> %rax
    movq   %rax, 8(%rdi, %rcx, 8)   # v2[i] = s1.vi[i]/4
    movq   ar2(%rbp), %rsi          # &ar2 -> %rsi
    movl   (%rsi, %rcx, 4), %ebx    # ar2[i] -> %ebx
    movl   %ebx, 4(%rdi, %rcx, 1)   # v3[i] = ar2[i]

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

finefor1:

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

#-------------------------------------------------------------------------------
.GLOBAL _ZN2cl5elab1EPc3st2                  # cl cl::elab1(char ar1[], st2 s2)
#-------------------------------------------------------------------------------
# activation frame:
# -----------------
#  cla.v1/v3    -88
#  cla.v2[0]    -80
#  cla.v2[1]    -72
#  cla.v2[2]    -64
#  cla.v2[3]    -56
#  s1 [MSB]     -48
#  s1 [LSB]     -40
#  i            -32
#  s2           -28
#  &ar1         -24
#  &this        -16
#  &indo        -8
#  %rbp         0
#-------------------------------------------------------------------------------
_ZN2cl5elab1EPc3st2:
# set stack locations labels:
```

```
    .set indo, -8
    .set this, -16
    .set ar1,  -24
    .set s2,   -28
    .set i,    -32
    .set s1,   -48
    .set cla,  -88

# prologue: activation frame:
    pushq %rbp
    movq  %rsp, %rbp
    subq  $88, %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)
    movl %ecx, 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:
    movslq i(%rbp), %rcx            # i -> %rcx
    movq   ar1(%rbp), %rdi          # &ar1 -> %rdi
    movb   (%rdi, %rcx, 1), %al     # ar1[i] -> %al
    movsbq %al, %rax                # ar1[i] -> %rax
    addq   %rcx, %rax               # ar1[i] + i -> %rax
    leaq   s1(%rbp), %rsi           # &s1 -> %rsi
    movl   %eax, (%rsi, %rcx, 4)    # s1.vi[i] = ar1[i] + i;

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

finefor2:

# cl cla(s1):
    leaq cla(%rbp), %rdi
    movq s1(%rbp), %rsi
    movq s1+8(%rbp), %rdx
    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
    movslq i(%rbp), %rcx            # i -> %rcx
    leaq   cla(%rbp), %rdi          # &cla -> %rdi
    leaq   s2(%rbp), %rsi           # &s2 -> %rsi
    movb   (%rsi, %rcx, 1), %al     # s2.vd[i] -> %al
    movb   %al, 4(%rdi, %rcx, 1)    # cla.v3[i] = s2.vd[i]

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

finefor3:

# copy cla into the memory space addressed by indo
    leaq cla(%rbp), %rsi            # source address
    movq indo(%rbp), %rdi           # destination address
    movabsq $5, %rcx                # repetitions
    rep movsq                       # execute transfer
```

```
# return intialized object address
    movq indo(%rbp), %rax

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

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