```
# EXTENSION 2016-09-20

##
# PRIMITIVES INTERRUPTS PINS.
##

    # load void reg() primitive interrupt handler in the IDT
    carica_gate TIPO_R     a_reg            LIV_UTENTE

# EXTENSION 2016-09-20

# SOLUTION 2016-09-20

    # load natl listen() primitive interrupt handler in the IDT
    carica_gate TIPO_LS    a_listen         LIV_UTENTE

    # load void broadcast(natl msg) interrupt primitive handler in the IDT
    carica_gate TIPO_B     a_broadcast      LIV_UTENTE

# SOLUTION 2016-09-20

# EXTENSION 2016-09-20

##
# PRIMITIVES INTERRUPTS HANDLERS.
##

#-----------------------------------------------------------------------------
.GLOBAL a_reg                           # void reg() primitive interrupt handler
#-----------------------------------------------------------------------------
a_reg:
    .cfi_startproc
    .cfi_def_cfa_offset 40
    .cfi_offset rip, -40
    .cfi_offset rsp, -16
    call c_reg                      # call C++ implementation
    iretq                           # return from interrupt
    .cfi_endproc

# EXTENSION 2016-09-20

# SOLUTION 2016-09-20

#-----------------------------------------------------------------------------
.GLOBAL a_listen                     # natl listen() primitive interrupt handler
#-----------------------------------------------------------------------------
# The listen() primitive will hang the calling process until the next broadcast
# message is sent. At the of the C++ implementation c_listen the calling process
# is placed in the global broadcast descriptor listeners queue and the scheduler
# is called. This is why we have to save the current process state (salva_stato)
# and load a new process (carica_stato).
#-----------------------------------------------------------------------------
a_listen:
    .cfi_startproc
    .cfi_def_cfa_offset 40
    .cfi_offset rip, -40
    .cfi_offset rsp, -16
    call salva_stato                # save current process state
    call c_listen                   # call C++ implementation
    call carica_stato               # load new process state
    iretq                           # return from interrupt
    .cfi_endproc

#-----------------------------------------------------------------------------
.GLOBAL a_broadcast       # void broadcast(natl msg) interrupt primitive handler
#-----------------------------------------------------------------------------
# The c_broadcast C++ implementation for this IDT subroutine will queue the
# broadcaster process in either the global broadcast descriptor broadcaster
# queue (there are still some listener processes which must call the listen()
# primitive to receive the broadcast message) or in the system ready process
```

```
# (all listener processes have received the broadcast message using the
# broadcast_all utility method). That's why we need to save the current process
# (broadcaster) process and load a new process state (the scheduler is called
# at the end of the C++ implementation).
#--------------------------------------------------------------------------------
a_broadcast:
    .cfi_startproc
    .cfi_def_cfa_offset 40
    .cfi_offset rip, -40
    .cfi_offset rsp, -16
    call salva_stato                # save current process state
    call c_broadcast                # call C++ implementation
    call carica_stato               # load new process state
    iretq                           # return from interrupt
    .cfi_endproc

# SOLUTION 2016-09-20
```