Aluno: Wendel Caio Moro
Aluno: João Victor Frans Pondaco Winandy

```
.set UART_rx_irq,0x08
.set UART_tx_irq,0x10
.set UART_tx_bfr_empty,0x40

# save registers
        lui   $k0, %hi(_uart_buff)      # get buffer's address
        ori   $k0, $k0, %lo(_uart_buff)

        sw    $a0, 5*4($k0)             # save registers $a0,$a1, others?
        sw    $a1, 6*4($k0)
        sw    $a2, 7*4($k0)
        sw    $a3, 8*4($k0)

# replace $xx for the apropriate registers

        lui    $a0, %hi(HW_uart_addr)# get device's address
        ori    $a0, $a0, %lo(HW_uart_addr)

# your code goes here
        lw    $k1, USTAT($a0)          # Read status

        lw     $a1, UINTER($a0)       # Read interrupt register
        ori    $a1, $a1, U_rx_irq     # remove interrupt request
        sw     $a1, UINTER($a0)

        and   $a1, $k1, $a1           # Is this reception?
        beq   $a1, $zero, tr          #   no, see if is transmission

#-------------------------------------------------
# handle reception
UARTrec:
        lui    $a2, %hi(Ud)           #get device adress
        ori    $a2, $a2, %lo(Ud)

        lw    $a1, NRX($a2)   # a1 = number of elements in queue
        nop
        addi  $a1, $a1, 1             # nrx++
        sw    $a1, NRX($a2)           # save new valor of nrx

        lw    $a1, RXTL($a2)          # a1 = tail index
        lw    $k1, UDATA($a0)                 # Read data from device

        addi  $a0, $a2, RX_Q          # get queue adress from Ud
        add   $a0, $a0, $a1           # a0 = rxq[rxtl]
        sb    $k1, 0($a0)             # save new character in queue
        addi  $a1, $a1, 1             # rx_tl++
        andi  $a1, $a1, 0xf           # rx_tl % 16
        sw    $a1, RXTL($a2)          # save new valor of rx_tl in Ud


#-------------------------------------------------
# return
_return:
```

```
        # restore registers
        lw    $a3, 8*4($k0)
        lw    $a2, 7*4($k0)
        lw    $a1, 6*4($k0)             # restore registers $a0,$a1, others?
        lw    $a0, 5*4($k0)

        eret                            # Return from interrupt



#-------------------------------------------------
# handle transmission
tr:
        lui   $a0, %hi(HW_uart_addr)# get device's address
        ori   $a0, $a0, %lo(HW_uart_addr)

        lw    $a1, UINTER($a0)          # Read interrupt register
        ori    $a1, $a1, U_tx_irq       # remove interrupt request
        sw    $a1, UINTER($a0)

        and   $a1, $k1, $a1             # Is this transmission?
        beq   $a1, $zero, _return       #   no, ignore it and return

        lui   $a2, %hi(Ud)              # get Ud adress
        ori   $a2, $a2, %lo(Ud)

        lw    $a1, NTX($a2)             # load NTX valor
        li     $k1, 16                  # a2 = 16

        beq   $a1, $k1, _return         # if transmission queue full, return

#-------------------------------------------------
pt2:
        lw    $a1, NTX($a2)             # a1 receive number of empty spaces in queue
        nop
        addi  $a1, $a1, 1               # ntx++
        sw    $a1, NTX($a2)             #save ntx in Ud

        lw    $a1, TXHD($a2)            # a1 = tail index
        addi  $a3, $a2, TX_Q            # a3 = adress of transmission queue
        add   $a3, $a3, $a1             # a3 = txq[txtl]

        lbu   $k1, 0($a3)               # get character from queue
        nop
        sb    $k1, UDATA($a0)           #save character to REMOTE

        addi  $a1, $a1, 1               # rx_hd++
andi  $a1, $a1, 0xf             # rx_hd % 16
sw    $a1, TXHD($a2)#save new valor of head index

j     _return
nop
```