```
/*
 *
 * author(s):   Michael Kohler - 11-108-289
 *               Lars SchÃ¼tz
 * modified:    2012-05-02
 *
 */

.include "nios_macros.s"
.include "address_map.s"

/* definiere einige Konstanten */
.equ WANDER_LEFT, 0
.equ WANDER_RIGHT, 1
.equ NORMAL_SPEED, 0x800    # je grÃ¶sser die
.equ FAST_SPEED, 0x300      # Zahl, desto gÃ¶s
ser
.equ SLOW_SPEED, 0xd00      # das Intervall

/*********************************************
**********************************
 * TEXT SECTION
 */
.text

/*********************************************
**********************************
 * Entry point.
 */
.global _start
_start:
        /* set up sp and fp */
        movia   sp, 0x007FFFFC
# stack starts from largest memory address
        mov             fp, sp

        /* This program exercises a few featur
es of the DE1 basic computer.
         *
         * It performs the following:
         *     1. displays a red light wanderi
ng from LEDR0 to LEDR9 and back again (and so
on...)
         *     2. speed of light can be increa
sed by KEY3, decreased by KEY1 and initial val
ue can be restored by KEY2
         */

        /* set up timer interval = 0x0000C350
steps * 1/(50 MHz) = 1 millisecond*/
        movia   r15, TIMER_COUNTER_LOW
        movui   r16, 0xC350
        sthio           r16, 0(r15)

        movia   r15, TIMER_COUNTER_HIGH
        movui   r16, 0x0000
        sthio           r16, 0(r15)

        /* start interval timer, enable its in
terrupts */
        movia   r15, TIMER_STOP_START_CONT_ITO
        movi            r16, 0b0111
# START = 1, CONT = 1, ITO = 1
        sthio           r16, 0(r15)

        /* enable pushbutton interrupts */
        movia   r16, PUSHBUTTON_BASE
        movi            r15, 0b01110
# set all 3 interrupt mask bits to 1 (bit 0 is
 Nios II Reset)
        stwio           r15, 8(r16)

        /* enable processor interrupts */
        movi            r16, 0b011
        # enable interrupts for timer and push
buttons
        wrctl           ienable, r16
        movi            r16, 1
        wrctl           status, r16


        /* Task (c) implemented wandering ligh
t */

    PRE:    # erstelle Register fÃ¼r die roten
 LEDs
            # r12 zeigt die Zahl
            # r13 zeigt die Richtung

    movi r12, 0x1
    movia r13, WANDER_LEFT
    movia r14, RED_LED_BASE
    movia r20, SPEED
    movia r21, SLOW_SPEED
    stw r21, 0(r20)


    MAIN:
    BUTTON_CHECK:   # key_pressed: zeige die g
edrÃ¼ckten Keys
                    # update_check: prÃ¼ft ob
ein Update erforderlich ist

    movia r20, KEY_PRESSED
    ldw r20, 0(r20)
    beq r0, r20, UPDATE_CHECK
    movi r21, KEY1
    beq r21, r20, PRESSED_KEY1
    movi r21, KEY2
    beq r21, r20, PRESSED_KEY2
    br PRESSED_KEY3

    PRESSED_KEY1:
    movia r20, SPEED
    movia r21, FAST_SPEED
    stw r21, 0(r20)
    br UPDATE_CHECK

    PRESSED_KEY2:
    movia r20, SPEED
    movia r21, SLOW_SPEED
    stw r21, 0(r20)
```

```
br UPDATE_CHECK

PRESSED_KEY3:
movia r20, SPEED
movia r21, SLOW_SPEED
stw r21, 0(r20)

UPDATE_CHECK:
movia r21, SPEED
ldw r20, 0(r21)
movia r22, TIME
ldw r21, 0(r22)
bgt r20, r21, END_MAIN


DO_GREEN_BLINKING:
call BLINK_GREEN

DO_RED_WANDERING:
stwio r12, 0(r14)
movia r20, WANDER_RIGHT
beq r13, r20, SHIFT_RIGHT

SHIFT_LEFT:
slli r12, r12, 0x1
movi r20, 0x1ff
bgtu r12, r20, SET_TO_RIGHT
br RESET_TIMER

SHIFT_RIGHT:
srli r12, r12, 0x1
movi r20, 0x2
blt r12, r20, SET_TO_LEFT
br RESET_TIMER

SET_TO_RIGHT:
movia r13, WANDER_RIGHT
br RESET_TIMER

SET_TO_LEFT:
movia r13, WANDER_LEFT

RESET_TIMER:
stw zero, 0(r22)

END_MAIN:
br MAIN


# Subroutine fÃ¼r grÃ¼n
BLINK_GREEN:
subi sp, sp, 8
stw r21, 0(sp)
stw r22, 4(sp)

# setze grÃ¼ne LEDs und speichere in r21
movia r22, GREEN_LED_BASE
ldwio r21, 0(r22)
bgt r21, zero, OFF

ON:
    movi r21, 0xff
    stwio r21, 0(r22)
    br END_BLINK_GREEN

    OFF:
    stwio zero, 0(r22)

    END_BLINK_GREEN:
    ldw r21, 0(sp)
    ldw r22, 4(sp)
    addi sp, sp, 8
    ret


/*****************************************
**********************************
 * DATA SECTION
 */
.data

/* to count how much time has passed*/
.global TIME
TIME:
        .word 0

.global KEY_PRESSED

    KEY_PRESSED:
    .word 0

    SPEED:
    .word 0

.end
```

```
/*
 *
 * author(s):   Michael Kohler - 11-108-289
 *              Lars SchÃ¼tz
 * modified:    2012-05-02
 *
 */

.include "nios_macros.s"
.include "address_map.s"
.extern KEY_PRESSED
.global PUSHBUTTON_ISR

/*****************************************
*****************************
 * Pushbutton - Interrupt Service Routine

 *

 *****************************************
*******************************/

PUSHBUTTON_ISR: /* speichere die Register auf
dem Stack */
    subi sp, sp, 12
    stw r10, 0(sp)
    stw r11, 4(sp)
    stw r12, 8(sp)

/* Task (d) managed speed of wandering light *
/

    # Unterbrechungen
    movia r10, PUSHBUTTON_BASE
    ldwio r11, 0xC(r10)
    stwio zero, 0xC(r10)

# zeige key_pressed welcher Key gedrÃ¼ckt wurd
e
CHECK_KEY1:
    andi r12, r11, 0b0010
    beq r12, zero, CHECK_KEY2
    movi r12, KEY1
    movia r10, KEY_PRESSED
    stw r12, 0(r10)
    br END_PUSHBUTTON_ISR

CHECK_KEY2:
    andi r12, r11, 0b0100
    beq r12, zero, CHECK_KEY3
    movi r12, KEY2
    movia r10, KEY_PRESSED
    stw r12, 0(r10)
    br END_PUSHBUTTON_ISR

CHECK_KEY3:
    andi r12, r11, 0b1000
    beq r12, zero, END_PUSHBUTTON_ISR
    movi r12, KEY3
    movia r10, KEY_PRESSED
    stw r12, 0(r10)

#speichere zurÃ¼ck ins Register
END_PUSHBUTTON_ISR:
    ldw r10, 0(sp)
    ldw r11, 4(sp)
    ldw r12, 8(sp)
    addi sp, sp, 12
    ret

.end
```