Rechnerarchitektur Serie 5

Dominik Bodenmann 08-103-053
Orlando Signer 12-119-715

13. Mai 2014

1 Theorie-Teil

1.1 Aufgabe 1

2 Praktischer Teil

Listing 1: knightRider.s

```
/* TODO: Task (b) Please fill in the following lines, then remove this line.
 *
 * author(s):   Dominik Bodenmann
 *       Orlando Signer
 * modified:    2014-05-09
 *
 */

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

/*****************************************************************************
 * 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 features of the DE1 basic computer.
   *
   * It performs the following:
   *     1. displays a red light wandering from LEDR0 to LEDR9 and back again
        (and so on...)
   *     2. speed of light can be increased by KEY3, decreased by KEY1 and
       initial value 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 interrupts */
  movia r15, TIMER_STOP_START_CONT_ITO
  movi    r16, 0b0111   # START = 1, CONT = 1, ITO = 1
  sthio   r16, 0(r15)
```

```
46
47  /* enable pushbutton interrupts */
48  movia   r16, PUSHBUTTON_BASE
49  movi    r15, 0b01110    # set all 3 interrupt mask bits to 1 (bit 0 is Nios
        II Reset)
50  stwio   r15, 8(r16)
51
52  /* enable processor interrupts */
53  movi    r16, 0b011      # enable interrupts for timer and pushbuttons
54  wrctl   ienable, r16
55  movi    r16, 1
56  wrctl   status, r16
57
58  /* r16 holds the value for the blinking LED (the position of the ball)
59   * r17 holds the current phase (1=Init 3=Play game 4=Finished)
60   * r18 is used to keep track of the direction (if 1, go up, else down
61   * r19, r20 are variables for free usage (no global usage)
62   * r21 stores the amount of time we want to wait each step (speed of the
         ball)
63   * r22 stores the points for player 1 and 2 (bit 0 to 3 for player 1, bit 4
          to 7 for player 2)
64   * r23 stores the button presses (0b1000 k3 pressed, 0b0010 k1 pressed, 0
         b1010 both pressed)
65   */
66
67  /* Initialize first red LED (light up) */
68  INIT:
69  movia   r15, RED_LED_BASE
70  movi    r16, 0x1    # Code for first LED
71  movi    r18, 0x0    # Direction bit (will be inverted first)
72  movi    r17, 0x1    # Set init phase
73  movi    r21, 0xF4   # How long to wait each step (0xF4 = 250)
74  movi    r22, 0x0    # reset score
75  call    SHOW_SCORE
76  br      CHECK_PHASE
77
78  CHECK_PHASE:
79  movi    r19, 0x1
80  beq     r17, r19, INIT_GAME     # state 1: init
81  movi    r19, 0x3
82  beq     r17, r19, PLAY_GAME     # state 3: playing
83  movi    r19, 0x4
84  beq     r17, r19, FINISHED_GAME   # state 4: finished
85  br      INIT            # if something is messed up, go back to init phase
86
87  /* Init phase: blinking LEDR4 and LEDR5 until both players press KEY1 and
        KEY3 */
88  INIT_GAME:
89  movi    r19, 0b1010
90  beq     r23, r19, START_GAME    # if k1 and k3 are pressed, start the game
91  movi    r19, 0x10         # Store value for L4
92  beq     r16, r19, SHOW_L5     # Check if only L4 is active. SHOW_L5 if true
93  br      SHOW_L4           # Else jump to SHOW_L4
94
```

```
95  SHOW_L4:
96  movi    r16, 0x10         # Store value for L4
97  stwio   r16, 0(r15)         # Display LEDs
98  br      DELAY           # Jump to delay
99
100 SHOW_L5:
101 movi    r16, 0x20         # Store value for L5
102 stwio   r16, 0(r15)         # Display LEDs
103 br      DELAY           # Jump to delay
104
105 /* Start the game */
106 START_GAME:
107 movi    r21, 0xF4         # Reset game speed
108 movia   r19, TIME
109 ldwio   r20, 0(r19)         # Get the Time from the Counter
110 andi    r18, r20, 0x1       # Set the direction
111 movi    r16, 0x10         # set the ball to the 4th led
112 movi    r17, 0x3          # Set phase 3
113 movi    r23, 0x0          # reset pressed buttons
114 call    SHOW_SCORE
115 br      PLAY_GAME
116
117 PLAY_GAME:
118 br      DO_DISPLAY_1
119
120 DO_DISPLAY_1:
121 movi    r19, 0x1
122 beq     r16, r19, LOWER_BORDER    # Check if we hit lower border
123 movi    r19, 0x200
124 beq     r16, r19, UPPER_BORDER    # Check if we hit upper border
125 movi    r19, 0b10
126 beq     r19, r23, PLAYER_1_FAIL   # No border, but player 1 pressed button
127 movi    r19, 0b1000
128 beq     r19, r23, PLAYER_2_FAIL   # No border, but player 2 pressed button
129 movi    r23, 0x0          # reset pressed buttons
130
131 DO_DISPLAY_2:
132 movi    r19, 0x1
133 beq     r18, r19, DO_DISPLAY_UP   # Check if we are going upwards (jump to
        DISPLAY_UP, if we are)
134 br      DO_DISPLAY_DOWN       # go To DISPLAY_DOWN if we're not going
        upwards
135
136 DO_DISPLAY_UP:
137 stwio   r16, 0(r15)         # Display current position
138 slli    r16, r16, 0x1       # Shift the value to the left (next LED)
139 br      DELAY           # Delay the next output
140
141 DO_DISPLAY_DOWN:
142 stwio   r16, 0(r15)         # Display current position
143 srai    r16, r16, 0x1       # Shift the value to the right (previous LED)
144 br      DELAY           # Delay the next output
145
146 DELAY:
```

```
147   movi    r23, 0x0          # reset all pressed buttons
148   br REAL_DELAY             # do the real delay
149
150   REAL_DELAY:
151   movia   r19, TIME
152   ldwio   r20, 0(r19)         # Get the Time from the Counter
153   blt     r20, r21, REAL_DELAY    # Check if we already waited more than (r21
          ) seconds
154   stwio   r0, 0(r19)          # Reset Time counter
155   br      CHECK_PHASE         # Go to display the next position
156
157   LOWER_BORDER:
158   andi    r19, r23, 0b0010      # check if k1 has pressed
159   movi    r20, 0b0010
160   beq     r19, r20, INVERT_DIRECTION  # player pressed button -> invert
          direction
161   br      PLAYER_1_FAIL       # player didnt press button
162
163   UPPER_BORDER:
164   andi    r19, r23, 0b1000      # check if k1 has pressed
165   movi    r20, 0b1000
166   beq     r19, r20, INVERT_DIRECTION  # player pressed button -> invert
          direction
167   br      PLAYER_2_FAIL       # player didnt press button
168
169   INVERT_DIRECTION:
170   subi    r21, r21, 0x10        # Subtract a fixed amount to the current
          waiting time
171   xori    r18, r18, 0x1       # Invert the direction after we hit the border
172   br      DO_DISPLAY_2
173
174   PLAYER_1_FAIL:
175   mov     r20, r22          # get score
176   andi    r20, r20, 0xF0        # get score from player 2
177   srli    r20, r20, 0x4       # shift right by 4 to get correct score
178   movi    r19, 0xA          # max score 10
179   bge     r20, r19, PLAYER_2_WIN    # if p2 reached 10 poins, he wins
180   addi    r22, r22, 0x10        # else add one point
181   br      START_GAME          # start the game
182
183   PLAYER_2_FAIL:
184   mov     r20, r22          # get score
185   andi    r20, r20, 0x0F        # get score from player 1
186   movi    r19, 0xA          # max score 10
187   bge     r20, r19, PLAYER_1_WIN    # if p1 reached 10 poins, he wins
188   addi    r22, r22, 0x01        # else add one point
189   br      START_GAME          # start the game
190
191   PLAYER_1_WIN:
192   movi    r17, 0x4          # set phase 4
193   movi    r16, 0b1111
194   stwio   r16, 0(r15)         # Display 4 leds for p1
195   br      DELAY
196
```

```
197
198   PLAYER_2_WIN:
199   movi    r17, 0x4          # set phase 4
200   movi    r16, 0b1111000000
201   stwio   r16, 0(r15)         # Display 4 leds for p2
202   br      DELAY
203
204   FINISHED_GAME:
205   andi    r19, r23, 0b0100      # check if k2 pressed
206   bgt     r19, r0, INIT       # init if k2 is pressed
207   br      DELAY
208
209   SHOW_SCORE:
210   mov     r12, ra           # store the return address, so we can later jump
          back
211   andi    r20, r22, 0xF0        # get score from player 2
212   srli    r20, r20, 0x4       # shift right by 4 to get correct score
213   call    LED_NUMBER
214   mov     r19, r10
215   slli    r19, r19, 24        # shift the number to the left
216
217   andi    r20, r22, 0x0F        # get score from player 1
218   call    LED_NUMBER
219   or      r19, r19, r10       # combine the 2 LED-numbers
220   movia     r20, HEX3_HEX0_BASE     # stores the hex base addres
221   stwio     r19, 0(r20)         # store the numbers to the LEDs
222   mov     ra, r12
223   ret
224
225
226   /* Gets the number from r20 and converts it into a LED-number and stores it
          in r10 */
227   LED_NUMBER:
228   movi    r10, 0b00111111       #Display value for 0
229   movi    r11, 0x0
230   beq     r20, r11, RETURN      #Check if score is 0
231
232   movi    r10, 0b00000110       #Display value for 1
233   movi    r11, 0x1
234   beq     r20, r11, RETURN      #Check if score is 1
235
236   movi    r10, 0b01011011       #Display value for 2
237   movi    r11, 0x2
238   beq     r20, r11, RETURN      #Check if score is 2
239
240   movi    r10, 0b01001111       #Display value for 3
241   movi    r11, 0x3
242   beq     r20, r11, RETURN      #Check if score is 3
243
244   movi    r10, 0b01100110       #Display value for 4
245   movi    r11, 0x4
246   beq     r20, r11, RETURN      #Check if score is 4
247
248   movi    r10, 0b01101101       #Display value for 5
```

```
249    movi    r11, 0x5
250    beq     r20, r11, RETURN      #Check if score is 5
251
252    movi    r10, 0b01111101       #Display value for 6
253    movi    r11, 0x6
254    beq     r20, r11, RETURN      #Check if score is 6
255
256    movi    r10, 0b00000111       #Display value for 7
257    movi    r11, 0x7
258    beq     r20, r11, RETURN      #Check if score is 7
259
260    movi    r10, 0b01111111       #Display value for 8
261    movi    r11, 0x8
262    beq     r20, r11, RETURN      #Check if score is 8
263
264    movi    r10, 0b01101111       #Display value for 9
265    movi    r11, 0x9
266    beq     r20, r11, RETURN      #Check if score is 9
267
268    RETURN:
269    ret
270  /*****************************************************************************
271   * DATA SECTION
272   */
273  .data
274
275  /* to count how much time has passed*/
276  .global TIME
277  TIME:
278    .word 0
279
280  /* TODO: Task (c) you may also want to add things here (but you don't need to
         ) */
281  .end
```

Listing 2: pushbutton.s

```
/* TODO: Task (b) Please fill in the following lines, then remove this line.
 *
 * author(s):   Dominik Bodenmann
 *        Orlando Signer
 * modified:    2014-05-09
 *
 */

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


/*****************************************************************************
 * Pushbutton - Interrupt Service Routine
 *
*****************************************************************************/
.global PUSHBUTTON_ISR
PUSHBUTTON_ISR:

  /* r19, r20 are variables for free usage (no global usage). */
  /* r23 stores the button presses (0b1000 k3 pressed, 0b0010 k1 pressed, 0
      b1010 both pressed) */


  movia   r19, PUSHBUTTON_BASE
  ldwio   r20, 0xC(r19)     # store pressed button in r20
  stwio   r0, 0xC(r19)      # clear the interrupt
  xor     r23, r23, r20     # store currently pressed button to r23
  ret

.end
```