Exercise sheet 4 – Hardware, Processor architecture

Goals:

- Interrupt handling

Exercise 4.1: Processor architecture: Interrupt handling (theoretical)

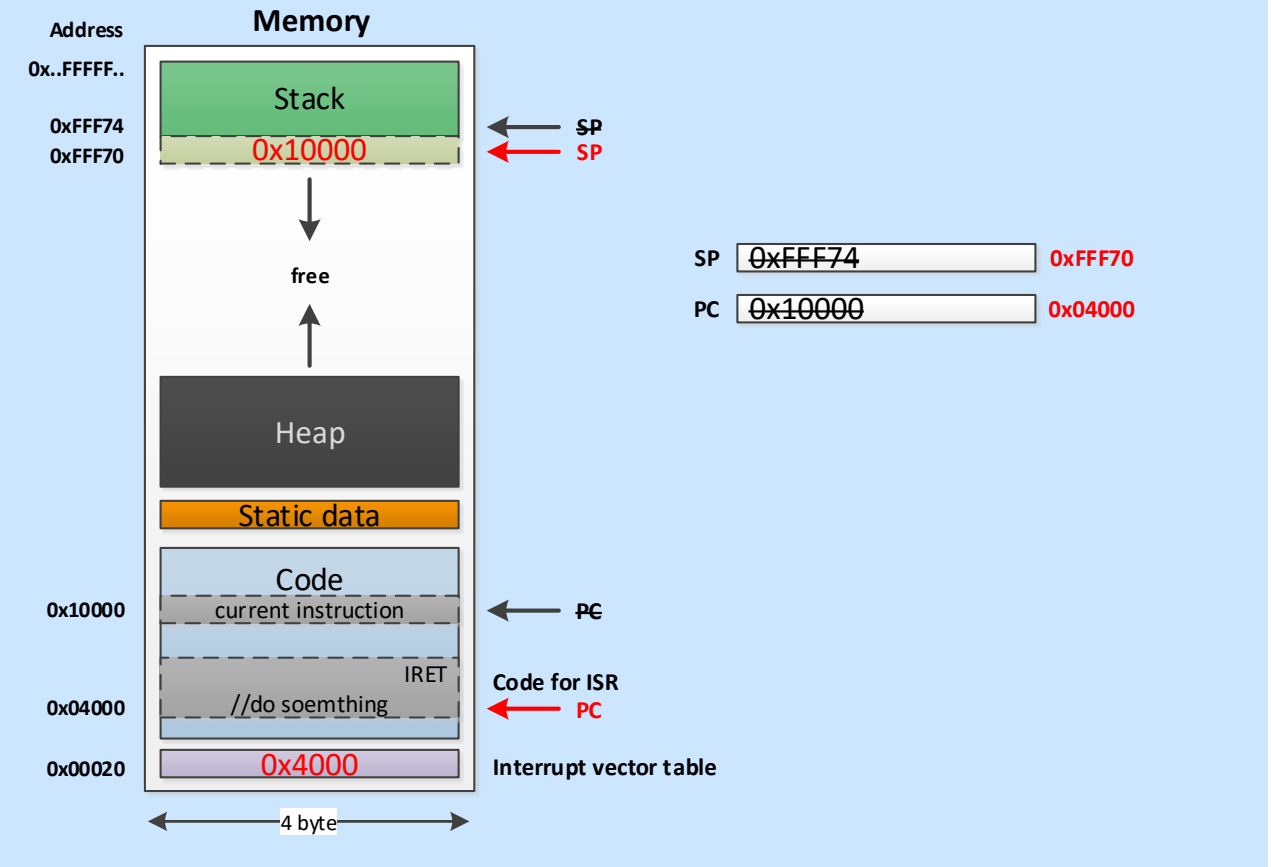
Given information:

- Interrupt vector address is 0x20
- Position of interrupt service routine (ISR) starts at 0x4000
- Stackpointer (SP) contains 0xFFF74
- Program counter (PC) respectively instruction pointer (IP) contains 0x10000
- Consider a micro controller without an operating system

(a) Recapitulate the sequence of an interrupt.

(b) Draw a sketch and show the changes according to the processing of an interrupt in different colours. The drawing should contain at least a memory view including addresses (32 bit: 4 byte with) and the PC and SP registers.

Proposal for solution:



Exercise 4.2: Short introduction into Ardiuno programming with Tinkercad circuits (coding)

We do this together, as a kick starter for you.

(a) Login into `https://www.tinkercad.com`. Use the *ad hoc* provided link (during the exercise) to login to the Tinkercad class room.

(b) Open circuits

(c) Create new circuit

(d) Open „Button“ example: *Starters Arduino: „Button“*

(e) Start simulation and then press the button

(f) Stop the simulation

(g) Inspect the code by changing to „Text“

Exercise 4.3: Update RA repository

(a) `cd RA_exercises`

(b) `git pull`

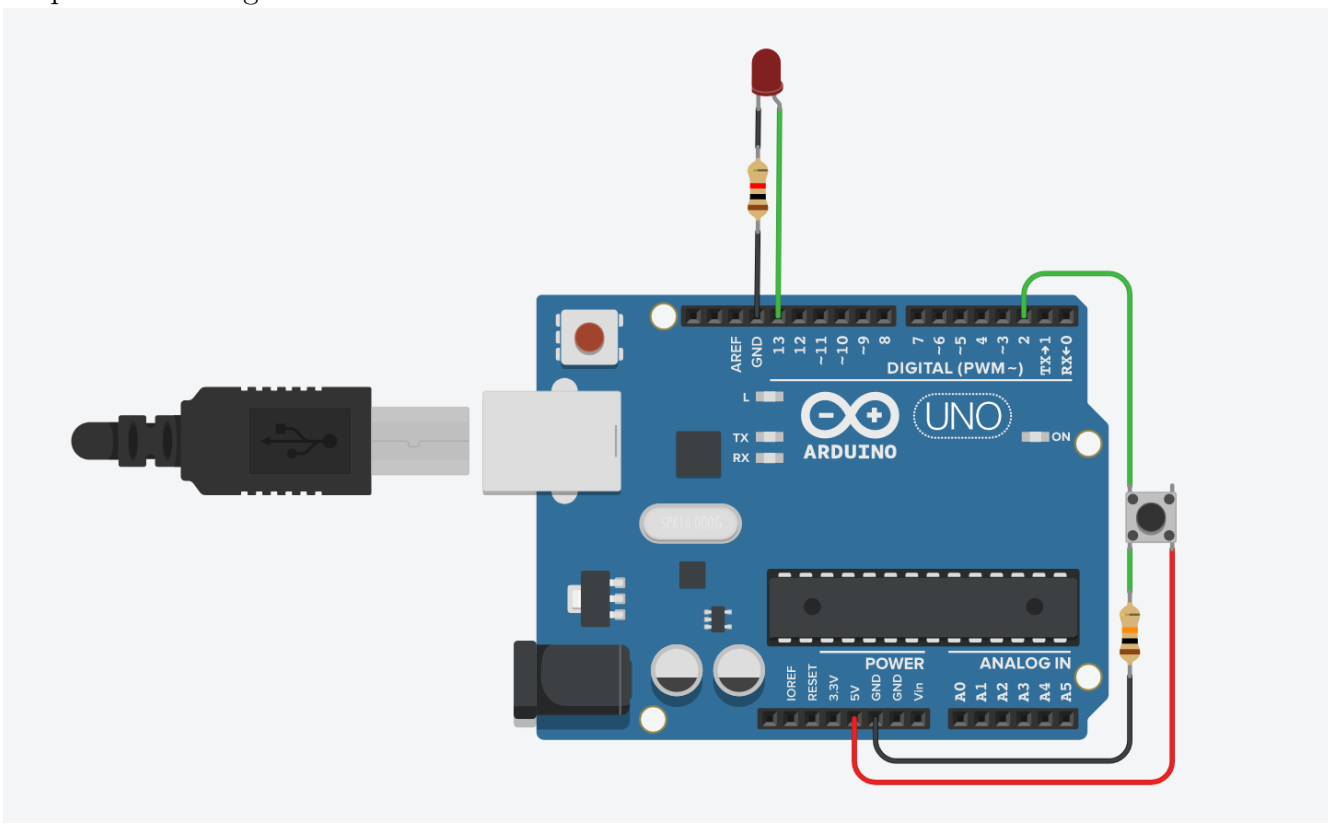
Exercise 4.4: Processor architecture: Interrupt handling (coding)

We want to write an Arduino sketch which toggles the built-in LED when a button is pressed. If the button is pressed, an interrupt occurs which calls an ISR.

(a) Use the *ad hoc* provided link (during the exercise) to login to the Tinkercad class room.

(b) Create a new circuit using the *Starters Arduino: „Button“*

(c) Prepare the wiring as follows:



(d) Copy the content of the `RA_exercises/sheet_04_online_only/io_interrupt/io_interrupt.ino` template into the code part of your Tinkercad circuit.

(e) Follow the TODOs in the code. Some configuration depends on your wiring of the I/O pins. *Hint: The Arduino reference contains descriptions of the used functions:* ***https://www.arduino.cc/reference/en.***

Proposal for solution:

```
1  //PIN configuration
2  const int BUTTON_PIN = 2;
3  const int LED_PIN    = LED_BUILTIN; //Internal built-in LED
4
5  //Global variables
6  volatile bool led_state = false;
7  volatile unsigned long time_prev_rising_edge = 0;
8
9  /*!
10  * setup() is called once on startup/reset of the Arduino
11  */
12 void setup(){
13   //Configure serial connection
14   Serial.begin(9600);
15
16   //configure PINS
17   pinMode(BUTTON_PIN, INPUT);
18   pinMode(LED_PIN,    OUTPUT);
19
20   //Switch LED initally off
21   digitalWrite(LED_PIN, false);
22
23   //Configure interrupt
24   attachInterrupt(digitalPinToInterrupt(BUTTON_PIN), isr_button_pressed, RISING);
25 }
26
27 /*!
28  * loop() is called as fast as possbile.
29  *
30  * As you can see, there is no call to a function
31  * changing the LED state in the main-loop
32 */
33 void loop(){
34   Serial.print("LED state is: ");
35   if(led_state) {
36     Serial.println("ON");
37   } else {
38     Serial.println("OFF");
39   }
40   delay(1000); //wait for 1 sec
41 }
42
43 /*!
44  * isr_button_pressed() = Interrupt service routine
45  * Change here the state of the LED
46  */
47 void isr_button_pressed(){ //interrupt ser
48   if (millis() - time_prev_rising_edge > 50) { //only react on rising edges every 50 m
49     led_state = !led_state;
50     digitalWrite(LED_PIN, led_state);
51
52     if(led_state) {
53       Serial.println("LED STATE UPDATED: ON");
```

```
54      } else {
55        Serial.println("LED STATE UPDATED: OFF");
56      }
57
58      time_prev_rising_edge = millis();
59    }
60  }
```

(f) Start the simulation

(g) Press the button to test your sketch. Does it work as expected?

(h) You can also open the „Serial Monitor“ to do some debugging with the text based logging.