CT Lab: Structured Programming in Assembly

1 Introduction

In this lab you will use structured programming techniques to implement an assembly program. Depending on the state of button T0, the potentiometer POT1 and the DIP switches S7..S0 are read and compared. The results are displayed in various ways on the 7-segment display and the LED31..0.

2 Learning Objectives

- You are able to structure a given problem using a structogram.
- You are able to convert a structogram containing sequences, selections and loops into a clear, easy-to-read assembly code.

3 Tasks 1

a) Draw a structogram with the following functionality and then implement it in assembly.

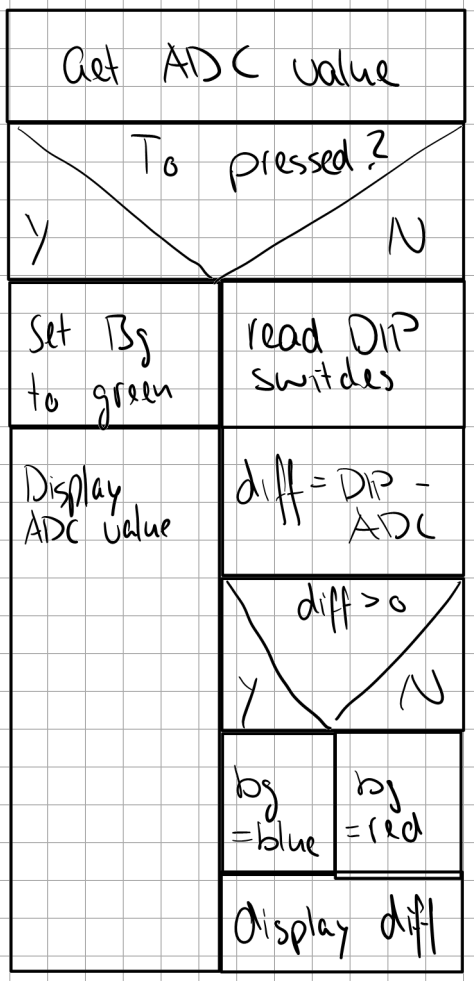
- The program shall read the **ADC-value** using the given function **`adc_get_value()`** and store the value in a register.
- The program shall check the state of button T0. **If T0 is pressed**
  - The background color of the LCD shall be set to **green** (case green) and the read ADC-value shall be displayed on the 7-segment display.

  Otherwise (i.e. T0 not pressed)
  - The program shall read an 8-bit value from DIP switches S7..S0.
  - The ADC-value shall be subtracted from the value read from the switches: **diff = value(S7..S0) – ADC-value**
  - If **diff >= 0** the background color of the LCD shall be set to **blue** (case blue). Otherwise the background color of the LCD shall be set to **red** (case red).
  - diff shall be displayed on the 7-segment display.

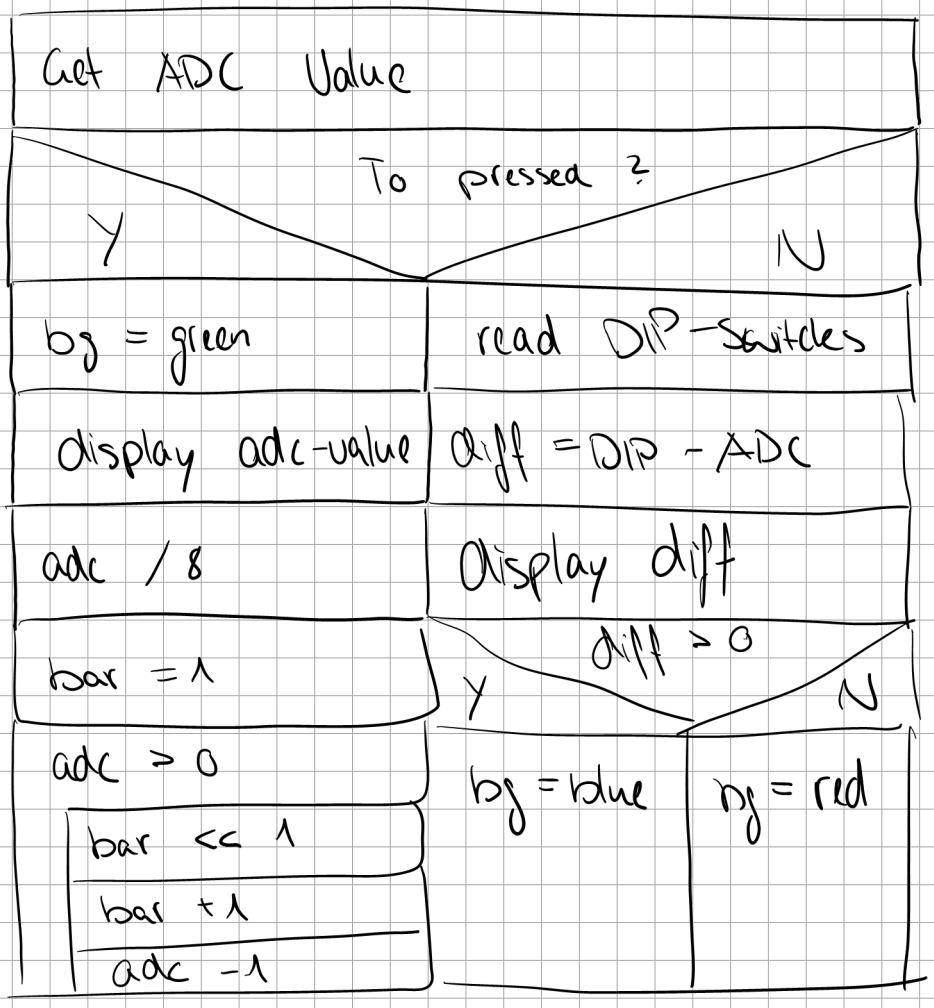
Verify that all cases are implemented correctly.

- ADC stands for Analog/Digital Converter. An ADC converts an analog value into a binary representation. In our case the voltage on the potentiometer is read (voltage divider). The voltage is represented as 8-bit value. A voltage of 0 Volt corresponds to an ADC-value of 0, whereas a voltage of 3.3 Volt corresponds to an ADC-value of 255d.
- The function call **`BL adc_get_value`** will return the ADC-value in the 32-bit register **`R0`**.
- There is a special register address available to output the content of a register as HEX on the 7-segment display. See '7-Segment Binary Interface' on ennis.zhaw.ch.

a)


Get ADC value
To pressed?
Y
N
Set Bg to green
read DIP switches
Display ADC value
diff = DIP - ADC
diff > 0
Y
N
bg =blue
bg =red
display diff

b)


Get ADC Value
To pressed ?
Y
N
bg = green
read DIP-Switches
display adc-value
diff = DIP - ADC
adc / 8
display diff
bar = 1
diff > 0
Y
N
adc > 0
bg = blue
bg = red
bar << 1
bar + 1
adc - 1

b) Extend your structogram as well as your assembly program for the case **green**:

- The program shall output an LED-bar on LED31..0. The LED-bar shall have a magnitude depending on the ADC-value. I.e.
  - ADC-values 0 .. 7d → LED0 is turned on
  - ADC-values 8d .. 15d → LED0 and LED1 are turned on
  - ADC-values 16d .. 23d → LED0, LED1 and LED2 are turned on
  - And so on up to 248d .. 255d → all LEDs are turned on

- Use a division by 8 (shift right) to scale the ADC-value from 8-bit down to 5-bit. Then use a loop to calculate the magnitude of the LED-bar.

c) Extend your structogram as well as your assembly program for the case **blue**:

- The program shall evaluate the previously calculated diff. In case diff can be represented with
  - 2 bit, i.e. diff < 4, display '2 Bit' on LCD
  - 4 bit, i.e. diff < 16, display '4 Bit' on LCD
  - 8 bit, i.e. in all other cases, display '8 Bit' on LCD

- Use the given function **`write_bit_ascii()`** to output the string ‚**Bit**‘ on the LCD. Why is it not possible to directly output the complete string (e.g. ‚**2 Bit**‘)?

d) Extend your structogram as well as your assembly program for the case **red**:

- The program shall count the number of binary zeros in diff.
- The counted number shall be output on the second line of the LCD.

- Update your structogram such that it is consistent with your implementation.
- Freely available tool to draw structograms: http://structorizer.fisch.lu/

4 Grading

The working programs have to be demonstrated. The individual students have to show understanding of their solution and their source code. Students have to be able to explain their solution.

Task	Criteria	Weight
1 a)	The program fulfills the defined functionality.	1/4
1 b)	The program fulfills the defined functionality.	1/4
1 c)	The program fulfills the defined functionality	1/4
1 d)	The program fulfills the defined functionality	1/4