Lab Sheet: Inline Assembler

The goal of this lab sheet is to exercise basic usage of inline assembler code, using the GNU toolchain, in particular `gcc`. The basic background information is in the Tutorial on Inline Assembler.

Resources and Infrastructure

- Tutorial on Inline Assembler
- ARM inline assembly blog
- GCC Manual, Section "Extended Asm"
- Sample sources: `sample0.c`
- Sample sources: `sumav1_asm.c`
- Sample sources: `sumav3_asm.c`

It is recommended that you start with downloading Sample sources: `sample0.c`, compile it and run it, as described in the header of the file. You can gain deeper insight in the inter-play between C and assembler by using `gdb` and focussing at the interface between C and assembler code.

Step 1: Simple example: sum-of-pair

Continuing the second example from the inline-assembler tutorial, the task here is to implement a function that takes a pair, as defined in the `sumav1_asm.c` sample source code, as input and to compute and return the sum of the components in the pair, using inline assembler. The interface for the function should look like this:

```
ulong sumpair_asm(pair_t *pair);
```

Step 2: Another example: `strlen` (Week 8/9)

As a second example, the goal is to write inline assembler code, that takes a string as input, and calculates the length of the string. Recall that strings in C are represented as arrays of `char` and each `char` has a size of 1 byte. Make sure to use the correct assembler-load instruction to get the next character. Strings are terminated by the `\0` (the value 0, not the symbol 0), which doesn't count as part of the string. To test the result, you can compare it with the result of the libc function `strlen` or with the C function presented in an earlier class.

Hint: If you struggle with the assembler coding required for this exercise, look-up the screencast on assembler programming of `strcpy`.

Step 3: Return to the button-controlled LED (Week 8/9)

Finally, return to the lab-sheet on the LED and Button (from Weeks 3/5), and **implement the code for controlling the LED and the button in inline Assembler**. Start with the C code that you have produced to set the function select register and to read from a button and write to an LED device. Now, modify this code so that the control of the LED is done from inline Assembler code in the program. Check the sample sources section for sample source code on how to turn on the on-board LED. You should try this step only after having done the first two steps in this lab on inline Assembler.