

1DT301 – Lab assignment 3

Goal for this lab:

- Learn to access data stored in .data section by using PC relative addressing
- Learn to write simple subroutines
- Learn how to connect a pushbutton and read input values

Presentation of results:

Presentation of results:

- 1) You must submit a report for assignment. The source code and makefiles should be uploaded to Gitlab, or a similar service. The report and source code must be submitted on Moodle before the deadline.
- 2) You have two options for the report:
 - a. Write a report in LaTeX as in Lab1, submit the report as .pdf and give a link to Gitlab, either in the report or on Moodle.
 - b. Write the report in Markdown, as an .MD file included in your gitlab repo and submit the Gitlab link on Moodle.
- 3) To pass the assignment, you must pass all tasks, submit working code as a Gitlab link and write a well-structured report. Only submitting code is not enough to pass the assignment!

Tasks

Task 1:

Write a program to calculate the average value of 8 numbers defined in the .data section. Then, show the result in the terminal (i.e. Minicom). You can use the following code as a template:

```
.thumb_func      @ Necessary because sdk uses BLX
.global main     @ Provide program starting address to linker

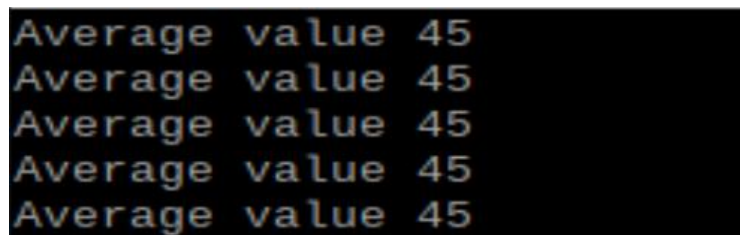
main:
    BL  stdio_init_all @ initialize uart or usb
loop:
    LDR R0, =my_array
    MOV R1, #8        @ 8 elements in the array
    BL  average        @ Call the subroutine average, with parameters R0 and R1

    @Print string and average value
    MOV R1, R0        @ Move average value to printf parameter R1
    LDR R0, =message_str @ load address of helloworld string
    BL  printf        @ Call pico_printf
    B   loop          @ loop forever

@Subroutine average takes the parameters:
@R0 - Memory address to first element of integer array
@R1 - Number of integers in the array
@R0 - Return value (integer average value)
average:
    @Your task is to implement this subroutine

.data
.align 4 @ necessary alignment
message_str: .asciz "Average value %d\n"
.align 4 @ necessary alignment
my_array: .word 10, 20, 30, 40, 50, 60, 70, 80
```

To show the result, the easiest way is probably to write the result continuously in an infinite loop, as in the Hello World program:



```
Average value 45
Average value 45
Average value 45
Average value 45
Average value 45
```

Remark: There is no division operation in the ARMv6m, but you can assume that the number of integers in the array is a power of two!

You **must** implement the average calculation as a subroutine!

Task 2:

Connect two LEDs to GP0+GP1 and two pushbuttons: One connected to GP2 and one to GP3.

The pushbuttons look like this:



First, you must find out how to connect the push button and how to set up the pin to read if it is pushed down or not!

Then, write a program with the following functionality:

If push button on GP1 is pushed down, turn on both LEDs. If push button on GP2 is pushed down, turn off both LEDs. Ignore the case when both buttons are down!

In this task, **you must use a C function** to read the pin!

Hint: You can use a C function in the RPi Pico SDK to read the input that works almost like the `gpio_put()` function. Search the SDK documentation to find out about this function. Remember to add it to the `sdlink.c` file because the function is an inline function!

Task 3:

Use the same hardware setup as in Task 2.

This time, **you are not allowed to use C functions** to read from the input pin or to turn the LEDs on/off! You must use read and write instructions to and from hardware registers. You can use the example from the book, chapter 9.

You **are allowed to use C functions to initialize and set direction**, like in the program listing 8-1 in the book. Alternatively, you can use the `gpioinit` function in Listing 9-5 to initialize the pins.