Linnaeus University

1DT301 - Computer Technology 1
Assignment 3

Group number: Group I
Group member: Jiaxin Wang, Zejian Wang
Student ID: jw223jc@student.lnu.se
zw222bb@student.lnu.se

Task 1

```
@
@ Showing to Tomas on October 24th, 2022
@ A program to calculate the average number of 8 numbers by
@ subroutine
@
.thumb_func          @ Necessary because sdk uses BLX
.global main         @ Provide program starting address to linker


main:
        BL stdio_init_all       @ intialize uart or usb
loop:
        LDR R0, =my_array
        MOV R1, #8              @ 8 elements in the array
        BL average              @ Call the subroutine average, with param
                                 eters r0 and r1


        @Print string and average value
        MOV R1, R0          @ Move average value to printf parameter 1
        LDR R0, =message_str  @ load address of message_str
        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 integer in the array
@R0 - Return value (integer average value)

average:
        PUSH  {lr}          @ push the link register
        BL sum              @ branch to sum
        LSR r0, r2, #3      @ shift right the value in R2 3 bits
        POP {pc}            @ pop the program counter
        BX lr
sum:
        SUB r1, r1, #1      @ minus 1 from R1
        LSL r3, r1, #2      @ shift left the value in R1 2 bits
        LDR r3, [r0, r3]    @ load the value to R3
        ADD r2, r2, r3      @ add the value in R3 to R2
        CMP r1, #0          @ compare the value in R1 with 0
        BNE sum             @if not equal, branch to sum again
        BX lr

.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
```

Task 2

```
@
@ Showing to Tomas on October 24th, 2022
@ Assembler program to control a LED by pressing pushbutton
@ connected to the Raspberry Pi Pico GPIO port using the Pico
@ SDK.
@

        .EQU    LED_PIN1, 0
        .EQU    BUTTON_1, 1
        .EQU    BUTTON_2, 2
        .EQU    GPIO_OUT, 1
        .EQU    GPIO_IN, 0

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

main:
        MOV     R0, #LED_PIN1       @ initialize LED as output
        BL      gpio_init
        MOV     R0, #LED_PIN1
        MOV     R1, #GPIO_OUT
        BL      link_gpio_set_dir

        MOV     R0, #BUTTON_1       @ initialize button 1 as input
        BL      gpio_init
        MOV     R0, #BUTTON_1
        MOV     R1, #GPIO_IN
        BL      link_gpio_set_dir


        MOV     R0, #BUTTON_2       @ initialize button 2 as input
        BL      gpio_init
        MOV     R0, #BUTTON_2
        MOV     R1, #GPIO_IN
        BL      link_gpio_set_dir


button_1:
        MOV R0, #BUTTON_1
        BL link_gpio_get                @get the state of button1
        CMP R0, #1                      @compare the state with 1
        BNE led_off         @if not equal to 1, then branch to led_off
        B button_2          @if equal to 1, then branch to button_2


button_2:
        MOV R0, #BUTTON_2
        BL link_gpio_get                @get the state of button2
        CMP R0, #1                      @compare the state with 1
        BNE led_on          @if not equal to 1, then branch to led_on
        B button_1          @if equal to 1, then branch to button_1


led_on:

        MOV     R0, #LED_PIN1                   @ turn on the LED
        MOV     R1, #1
        BL      link_gpio_put
        B   button_2                            @ branch to button_2
        BX  lr

led_off:

        MOV     R0, #LED_PIN1                   @ turn off the LED
        MOV     R1, #0
        BL      link_gpio_put
        B   button_1                            @ branch to button_1
        BX  lr
```

Task 3

```
@
@ Showing to Tomas on October 24th, 2022
@ Assembler program to control a LED and its blinking speed
@ by pressing push buttons connected to the Raspberry Pi Pico
@ GPIO port using the Pico SDK.
@

        .EQU    LED_PIN1, 0
        .EQU    BUTTON_1, 1
        .EQU    BUTTON_2, 2
        .EQU    BUTTON_3, 3
        .EQU    BUTTON_4, 4

        .EQU    GPIO_OUT, 1
        .EQU    GPIO_IN, 0

        .EQU    sleep_time1, 200     @ set time for a LED to blink faster
        .EQU    sleep_time2, 1000    @ set time for a LED to blink slower


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

main:
        MOV     R0, #LED_PIN1               @ initialize LED as output
        BL      gpio_init
        MOV     R0, #LED_PIN1
        MOV     R1, #GPIO_OUT
        BL      link_gpio_set_dir

        MOV     R0, #BUTTON_1             @ initialize button 1 as input
        BL      gpio_init
        MOV     R0, #BUTTON_1
        MOV     R1, #GPIO_IN
        BL      link_gpio_set_dir

        MOV     R0, #BUTTON_2           @ initialize button 2 as input

        BL      gpio_init
        MOV     R0, #BUTTON_2
        MOV     R1, #GPIO_IN
        BL      link_gpio_set_dir

        MOV     R0, #BUTTON_3           @ initialize button 3 as input
        BL      gpio_init
        MOV     R0, #BUTTON_3
        MOV     R1, #GPIO_IN
        BL      link_gpio_set_dir

        MOV     R0, #BUTTON_4           @ initialize button 4 as input
        BL      gpio_init
        MOV     R0, #BUTTON_4
        MOV     R1, #GPIO_IN
        BL      link_gpio_set_dir



led_off:
        MOV     R0, #LED_PIN1       @ turn off the led
        MOV     R1, #0
        BL      link_gpio_put

        MOV     R0, #BUTTON_1       @ get pressing status of BUTTON_1
        BL      link_gpio_get
        CMP R0, #0          @ check if BUTTON_1 is pressed or not
        BNE led_on          @ pressed, then go to led_on branch
        B led_off                       @ not pressed, then led will stay off,
                                        still go to led_off branch




led_on:

        MOV     R0, #LED_PIN1    @ turn on the led
        MOV     R1, #1
        BL      link_gpio_put

        MOV     R0, #BUTTON_2  @ get pressing status of BUTTON_2
        BL      link_gpio_get
        CMP R0, #0          @ check if BUTTON_2 is pressed or not
        BNE led_off         @ pressed, then go to branch led_off

        MOV     R0, #BUTTON_3  @ get pressing status of BUTTON_3
        BL      link_gpio_get
        CMP R0, #0             @ check if BUTTON_3 is pressed or not
        BNE blink_faster     @ pressed, then go to branch blink_faster

        MOV     R0, #BUTTON_4 @ get pressing status of BUTTON_4
        BL      link_gpio_get
        CMP R0, #0            @ check if BUTTON_4 is pressed or not
        BNE blink_slower @ pressed, then go to branch blink_slower

        B led_on            @ when button 2,3,4 are all not pressed, only
                             button 1 is pressed, then still go to led_on branch


blink_faster:
        MOV     R0, #LED_PIN1  @ turn on the led for 200 millisecs
        MOV     R1, #1
        BL      link_gpio_put

        LDR R0, =sleep_time1
        BL sleep_ms

        MOV     R0, #LED_PIN1   @ turn off the led for 200 millisecs
        MOV     R1, #0
        BL      link_gpio_put

        LDR R0, =sleep_time1
        BL sleep_ms

        MOV     R0, #BUTTON_2 @ get pressing status of BUTTON_2
        BL      link_gpio_get
        CMP R0, #0          @ check if BUTTON_2 is pressed or not
        BNE led_off         @ pressed, then go to branch led_off


        MOV     R0, #BUTTON_4 @ get pressing status of BUTTON_4
        BL      link_gpio_get
        CMP R0, #0          @ check if BUTTON_4 is pressed or not
        BNE blink_slower  @ pressed, go to blink_slower branch

        B blink_faster




blink_slower:
        MOV     R0, #LED_PIN1   @ turn on the led for 1000 millisecs
        MOV     R1, #1
        BL      link_gpio_put

        LDR R0, =sleep_time2
        BL sleep_ms
```

```
MOV     R0, #LED_PIN1  @ turn off the led for 1000 millisecs
MOV     R1, #0
BL      link_gpio_put

LDR R0, =sleep_time2
BL sleep_ms

MOV     R0, #BUTTON_2  @ get pressing status of BUTTON_2
BL      link_gpio_get
CMP R0, #0          @ check if BUTTON_2 is pressed or not
BNE led_off         @ pressed, then go to branch led_off


MOV     R0, #BUTTON_3  @ get pressing status of BUTTON_3
BL      link_gpio_get
CMP R0, #0          @ check if BUTTON_3 is pressed or not
BNE blink_faster  @ pressed, then go to branch blink_faster


B blink_slower
```