Task 1:

The code shown below is the completed version of task 1 (specifically the “average” subroutine is implemented). First, the return address is saved with “PUSH {LR}” to allow the average to be returned to main. We then use R2 and R3 to store the sum and index counter respectively. After this, we enter a loop (loop_sum) where r3 and r1 are compared (r1 = 8), to ensure that all elements have been accounted for, and if so (r3 = r1) the loop ends. We use the lines “MOV R5, R3”, “LSL R5, R5, #2”, and “LDR R4, [R0, R5]” to load the current array element- the first line copies the index from r3 into r5, the second shifts r5 by 2 bit to get the offset for the current element, and the third line loads the value from the address [r0 + r5] into r4. After this, “ADD R2, R2, R4” is used to add the current element (stored in r4) to r2 in order to update the sum, and finally “ADD R3, R3, #1” is used to increase the iterator. When the loop has ended, the total sum is moved from r2 to r0, and a logical shift right is performed (by three bits, dividing the sum by 8 to get the average). After all of this, POP {PC} is used to return control to “main”, with the average stored in r0.

```
.thumb_func
.global main
main:
    BL stdio_init_all

loop:
    LDR R0, =my_array     @ Load address of 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


average:
    PUSH {LR}

    MOV R2, #0
    MOV R3, #0

loop_sum:
    CMP R3, R1
    BEQ end_loop

    MOV R5, R3
    LSL R5, R5, #2
    LDR R4, [R0, R5]
    ADD R2, R2, R4
```

```
    ADD R3, R3, #1

    B loop_sum

end_loop:
    MOV R0, R2
    LSR R0, R0, #3

    POP {PC}
    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:

This task works to assign two buttons to on and off for the use of turning an LED on (or off) in C (as opposed to the next task in assembly). This is done by first initializing each button and LED to a GPIO

pin (with gpio_init() for each element), and then setting the LED as an output and buttons and inputs (with gpio_set_dir() for each element). After this, we have an infinite loop (while (true)) which specifies that if button B1 is pressed, the LED will be turned on (with gpio_put(LED, 1)), and if button B2 is pressed, the LED will be turned off (with gpio_put(LED, 0)).

```
#include "pico/stdlib.h"
#include "hardware/gpio.h"

void main() {

    const uint LED = 0;
    const uint B1 = 1;
    const uint B2 = 2;

    gpio_init(LED);
    gpio_set_dir(LED, GPIO_OUT);

    gpio_init(B1);
    gpio_set_dir(B1, GPIO_IN);
    gpio_pull_up(B1);

    gpio_init(B2);
    gpio_set_dir(B2, GPIO_IN);
    gpio_pull_up(B2);

    while (true) {
        if (!gpio_get(B1)) {
            gpio_put(LED, 1);
        }

        if (!gpio_get(B2)) {
            gpio_put(LED, 0);
        }
    }
}
```

Task 3:

This task does the exact same thing as task 2, but the control is done in assembly and initialization is done in C.

```
    .EQU SIO_BASE, 0xd0000000
    .EQU SIO_GPIO_IN_OFFSET, 0x04
    .EQU SIO_GPIO_OUT_SET_OFFSET, 0x14
    .EQU SIO_GPIO_OUT_CLR_OFFSET, 0x18
    .EQU LED_PIN, 0
    .EQU BUTTON1, 1
    .EQU BUTTON2, 2
    .EQU GPIO_OUT, 1
    .EQU GPIO_IN, 0

.thumb_func
.global main

main:
    MOV R0, #LED_PIN
    BL gpio_init
    MOV R0, #LED_PIN
    MOV R1, #GPIO_OUT
    BL link_gpio_set_dir

    MOV R0, #BUTTON1
    BL gpio_init
    MOV R0, #BUTTON1
    MOV R1, #GPIO_IN
    BL link_gpio_set_dir
    BL link_gpio_set_pull_up

    MOV R0, #BUTTON2
    BL gpio_init
    MOV R0, #BUTTON2
    MOV R1, #GPIO_IN
    BL link_gpio_set_dir
    BL link_gpio_set_pull_up

button1check:
    MOV R0, #BUTTON1
    LDR R1, =SIO_BASE
    LDR R2, [R1, #SIO_GPIO_IN_OFFSET]
    MOV R3, #1
    LSL R3, R0
    AND R2, R2, R3
    CMP R2, #0
    BNE button2check
    MOV R0, #1
    LDR R1, =SIO_BASE
    STR R0, [R1, #SIO_GPIO_OUT_SET_OFFSET]
    B button2check
button2check:
```

```
MOV R0, #BUTTON2
LDR R1, =SIO_BASE
LDR R2, [R1, #SIO_GPIO_IN_OFFSET]
MOV R3, #1
LSL R3, R0
AND R2, R2, R3
CMP R2, #0
BNE button1check
MOV R0, #1
LDR R1, =SIO_BASE
STR R0, [R1, #SIO_GPIO_OUT_CLR_OFFSET]
B button1check
```