Task1: 1a: This code sets a single LED as an output, with two buttons set as inputs. The main code consists of an infinite while loop which continuously checks if the “on” and “off” buttons are pressed, and when they are gpio_put(LED_PIN, 0/1) is used to adjust the LED to its correct state.

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

// Define GPIO pins
#define LED_PIN 0         // Pin connected to the LED
#define BUTTON_ON_PIN 1   // Pin connected to Button A (used to turn the
LED ON)
#define BUTTON_OFF_PIN 2 // Pin connected to Button B (used to turn the
LED OFF)

int main() {
    // Initialize standard input/output
    stdio_init_all();

    // Initialize LED pin and set it as an output
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, GPIO_OUT);

    // Initialize Button ON pin and set it as an input
    gpio_init(BUTTON_ON_PIN);
    gpio_set_dir(BUTTON_ON_PIN, GPIO_IN);

    // Initialize Button OFF pin and set it as an input
    gpio_init(BUTTON_OFF_PIN);
    gpio_set_dir(BUTTON_OFF_PIN, GPIO_IN);

    // Infinite loop to monitor button presses and control the LED
    while (1) {
        // If Button ON is pressed, turn on the LED
        if (gpio_get(BUTTON_ON_PIN)) {
            gpio_put(LED_PIN, 1);
        }

        // If Button OFF is pressed, turn off the LED
```

```
        if (gpio_get(BUTTON_OFF_PIN)) {
            gpio_put(LED_PIN, 0);
        }

        // Add a small delay to debounce the buttons
        sleep_ms(20);
    }

    return 0; // This line is never reached
}
```

1b: One button switches the LED on, while the other turns it off. The GPIO pins are configured for the LED and buttons, with direct register access managing button inputs and LED control. A loop continuously checks button states, updates the LED status, and incorporates a debounce delay for consistent input handling.

```c
#include "pico/stdlib.h"

// Define GPIO pins
#define LED_PIN 0               // Pin connected to the LED
#define BUTTON_TURN_ON 1        // Pin connected to the first switch (to
turn LED ON)
#define BUTTON_TURN_OFF 2       // Pin connected to the second switch (to
turn LED OFF)

// Define direct register access for GPIO input and output
#define SIO_REG_BASE 0xd0000000
#define GPIO_INPUT (*(volatile uint32_t*)(SIO_REG_BASE + 0x0004))       //
Read GPIO input states
#define GPIO_SET_OUTPUT (*(volatile uint32_t*)(SIO_REG_BASE + 0x0014)) //
Set GPIO output state
#define GPIO_CLEAR_OUTPUT (*(volatile uint32_t*)(SIO_REG_BASE + 0x0018))
// Clear GPIO output state

int main() {
   // Initialize GPIO pins
   gpio_init(LED_PIN);          // Initialize the LED pin
   gpio_init(BUTTON_TURN_ON);   // Initialize the button to turn LED on
   gpio_init(BUTTON_TURN_OFF);  // Initialize the button to turn LED off

   // Set the direction of GPIO pins
   gpio_set_dir(LED_PIN, GPIO_OUT);         // Set LED pin as output
   gpio_set_dir(BUTTON_TURN_ON, GPIO_IN);   // Set Button ON pin as input
   gpio_set_dir(BUTTON_TURN_OFF, GPIO_IN);  // Set Button OFF pin as input

   // Initialize state variables
   bool is_led_active = false;             // Tracks if the LED is
currently ON
   bool previous_turn_on_state = false;    // Tracks the previous state of
the Button ON
   bool previous_turn_off_state = false;   // Tracks the previous state of
the Button OFF

   // Infinite loop to monitor button presses and control the LED
   while (1) {
```

```
        // Read the current state of the buttons using direct GPIO input
        bool turn_on_button_pressed = GPIO_INPUT & (1 << BUTTON_TURN_ON);
        bool turn_off_button_pressed = GPIO_INPUT & (1 << BUTTON_TURN_OFF);

        // If Button ON is pressed and was not pressed previously, turn on
the LED
        if (turn_on_button_pressed && !previous_turn_on_state) {
            if (!is_led_active) {
                GPIO_SET_OUTPUT = (1 << LED_PIN); // Set the LED output
high
                is_led_active = true;            // Update the LED state to
active
            }
        }

        // If Button OFF is pressed and was not pressed previously, turn
off the LED
        if (turn_off_button_pressed && !previous_turn_off_state) {
            if (is_led_active) {
                GPIO_CLEAR_OUTPUT = (1 << LED_PIN); // Clear the LED output
                is_led_active = false;              // Update the LED state
to inactive
            }
        }

        // Update the previous state of buttons for edge detection
        previous_turn_on_state = turn_on_button_pressed;
        previous_turn_off_state = turn_off_button_pressed;

        // Add a delay to debounce the buttons
        sleep_ms(50);
    }
}
```

1c: This program controls two LEDs on a Raspberry Pi Pico using two buttons. Pressing Button 1 turns on both LEDs, while pressing Button 2 turns them off. The GPIO pins are set up as inputs for the buttons and outputs for the LEDs, with direct register access handling button state detection and LED control. A continuous loop

monitors button presses, adjusts the LED states, and includes a delay for stable input processing.

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

// Define GPIO pins for LEDs and inputs
#define LED_MAIN 0                // Main LED GPIO pin
#define LED_SECONDARY 6           // Secondary LED GPIO pin
#define INPUT_BUTTON1 1           // Input button 1 GPIO pin
#define INPUT_BUTTON2 2           // Input button 2 GPIO pin

// Base address for SIO (Single Instruction Operation) in RP2040
#define SIO_BASE_ADDRESS 0xd0000000

// Define GPIO control registers
#define GPIO_CURRENT_STATE (*(volatile uint32_t*)(SIO_BASE_ADDRESS +
0x0004)) // GPIO state register
#define GPIO_ENABLE_OUTPUT_MASK (*(volatile uint32_t*)(SIO_BASE_ADDRESS +
0x0014)) // Enable GPIO output register
#define GPIO_DISABLE_OUTPUT_MASK (*(volatile uint32_t*)(SIO_BASE_ADDRESS +
0x0018)) // Disable GPIO output register

int main() {
   // Initialize GPIO pins
   gpio_init(LED_MAIN);
   gpio_init(LED_SECONDARY);
   gpio_init(INPUT_BUTTON1);
   gpio_init(INPUT_BUTTON2);

   // Set direction for LEDs as output and buttons as input
   gpio_set_dir(LED_MAIN, GPIO_OUT);
   gpio_set_dir(LED_SECONDARY, GPIO_OUT);
   gpio_set_dir(INPUT_BUTTON1, GPIO_IN);
   gpio_set_dir(INPUT_BUTTON2, GPIO_IN);

   // Initialize LED state and previous button states
   bool leds_active = false;            // Tracks if LEDs are active
   bool prev_button1_state = false;    // Tracks previous state of button
1
   bool prev_button2_state = false;    // Tracks previous state of button
2
```

```
   while (1) {
       // Read the current state of the input buttons
       bool button1_active = GPIO_CURRENT_STATE & (1 << INPUT_BUTTON1); //
Check if button 1 is pressed
       bool button2_active = GPIO_CURRENT_STATE & (1 << INPUT_BUTTON2); //
Check if button 2 is pressed

       // Check for a press event on button 1
       if (button1_active && !prev_button1_state) {
           // If LEDs are inactive, enable both LEDs
           if (!leds_active) {
               GPIO_ENABLE_OUTPUT_MASK = (1 << LED_MAIN) | (1 <<
LED_SECONDARY);
               leds_active = true;
           }
       }

       // Check for a press event on button 2
       if (button2_active && !prev_button2_state) {
           // If LEDs are active, disable both LEDs
           if (leds_active) {
               GPIO_DISABLE_OUTPUT_MASK = (1 << LED_MAIN) | (1 <<
LED_SECONDARY);
               leds_active = false;
           }
       }

       // Update previous button states for the next loop iteration
       prev_button1_state = button1_active;
       prev_button2_state = button2_active;

       // Add a small delay to debounce button presses
       sleep_ms(50);
   }
}
```

Task2

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

#define LED1 1
#define LED2 2
#define LED3 3
#define LED4 4
#define BUTT_UP 5
#define BUTT_DOWN 6
#define DELAY 300

// Variables for button debounce
void gpio_callback(uint gpio, uint32_t events);
static uint32_t last_interrupt_inc = 0;
static uint32_t last_interrupt_dec = 0;

// Counter variable
int cntr = 0;

// Set LEDs based on counter value
void set_leds() {
    for (int j = 0; j < 4; j++) {
        gpio_put(LED1 + j, (cntr >> j) & 1);
    }
}

// Increment counter
void cnt_up() {
    if (cntr < 15) {
        cntr++;
        set_leds();
    }
}

// Decrement counter
void cnt_down() {
```

```
    if (cntr > 0) {
        cntr--;
        set_leds();
    }
}

// GPIO interrupt callback
void gpio_callback(uint gpio, uint32_t events) {
    uint32_t current_time = to_ms_since_boot(get_absolute_time());

    if (gpio == BUTT_UP) {
        // Debounce button
        if (current_time - last_interrupt_inc > DELAY) {
            cnt_up();
            last_interrupt_inc = current_time;
        }
    } else if (gpio == BUTT_DOWN) {
        // Debounce button
        if (current_time - last_interrupt_dec > DELAY) {
            cnt_down();
            last_interrupt_dec = current_time;
        }
    }
}

int main() {
    stdio_init_all();

    gpio_init(LED1);
    gpio_set_dir(LED1, GPIO_OUT);

    gpio_init(LED2);
    gpio_set_dir(LED2, GPIO_OUT);

    gpio_init(LED3);
    gpio_set_dir(LED3, GPIO_OUT);

    gpio_init(LED4);
    gpio_set_dir(LED4, GPIO_OUT);
```

```
    gpio_init(BUTT_UP);
    gpio_set_dir(BUTT_UP, GPIO_IN);
    gpio_pull_up(BUTT_UP);

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


    // Enable interrupts with the callback
    gpio_set_irq_enabled_with_callback(BUTT_UP, GPIO_IRQ_EDGE_FALL, true,
gpio_callback);
    gpio_set_irq_enabled_with_callback(BUTT_DOWN, GPIO_IRQ_EDGE_FALL, true,
gpio_callback);

    // Initialize LEDs
    set_leds();

    // Main loop
    while (1) {
        tight_loop_contents();
    }
}
```

Task 3

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

#define LED1 1
```

```
#define LED2 2
#define LED3 3
#define LED4 4
#define BUTT_RESET 0

void gpio_callback(uint gpio, uint32_t events);
int64_t timer_callback(alarm_id_t id, void *unused);

//counter
int cntr = 0;

//leds on counter value
void set_leds() {
   for(int j = 0; j<4; j++){
       gpio_put(LED1 + j, (cntr >> j) & 1);
   }
}

void cnt_up() {
   if (cntr < 15) {
       cntr ++;
       set_leds();
   }
}

void cnt_down(){
   if (cntr > 0) {
       cntr --;
       set_leds();
   }
}
//interrupt for resetting the button
void gpio_callback(uint gpio, uint32_t events) {
   if ( gpio == BUTT_RESET) {
       cntr = 0;
       set_leds();
   }
}

//timer
```

```
int64_t timer_callback(alarm_id_t id, void *unused) {
    if (cntr < 15) {
        cnt_up();
    }
    return 1000*1000;
}

int main() {
    stdio_init_all();

    gpio_init(LED1);
    gpio_set_dir(LED1,GPIO_OUT);

    gpio_init(LED2);
    gpio_set_dir(LED2,GPIO_OUT);

    gpio_init(LED3);
    gpio_set_dir(LED3,GPIO_OUT);

    gpio_init(LED4);
    gpio_set_dir(LED4,GPIO_OUT);

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

    gpio_set_irq_enabled_with_callback(BUTT_RESET,
GPIO_IRQ_EDGE_FALL,true,gpio_callback);
    add_alarm_in_us(1000*1000, timer_callback, NULL, true);

    set_leds();

    while (1) {
        tight_loop_contents();
    }

    }
```