```
/*
 * assembly.s
 *
 */

 @ DO NOT EDIT
      .syntax unified
    .text
    .global ASM_Main
    .thumb_func

@ DO NOT EDIT
vectors:
      .word 0x20002000
      .word ASM_Main + 1

@ DO NOT EDIT label ASM_Main
ASM_Main:

      @ Some code is given below for you to start with
      LDR R0, RCC_BASE          @ Enable clock for GPIOA and B by setting bit 17 and
18 in RCC_AHBENR
      LDR R1, [R0, #0x14]
      LDR R2, AHBENR_GPIOAB     @ AHBENR_GPIOAB is defined under LITERALS at the end
of the code
      ORRS R1, R1, R2
      STR R1, [R0, #0x14]

      LDR R0, GPIOA_BASE        @ Enable pull-up resistors for pushbuttons
      MOVS R1, #0b01010101
      STR R1, [R0, #0x0C]
      LDR R1, GPIOB_BASE        @ Set pins connected to LEDs to outputs
      LDR R2, MODER_OUTPUT
      STR R2, [R1, #0]
      MOVS R2, #0               @ NOTE: R2 will be dedicated to holding the value on
the LEDs

@ TODO: Add code, labels and logic for button checks and LED patterns

main_loop:
      B     button_check @Branch to check for button press

button_check:
      LDR   R3,  [R0, #0x10] @ Load IDR to check for SW0 press
      MOVS  R4,  #1
      ANDS  R3,  R3,  R4
      CMP   R3,  #0
      BEQ   led_counter_SWO

      LDR   R3,  [R0, #0x10] @ Load IDR to check for SW2 press
      MOVS  R4,  #4
      ANDS  R3,  R3,  R4
      CMP   R3,  #0
      BEQ   write_leds_SW2
```

```
      LDR   R3,  [R0, #0x10] @ Load IDR to check for SW3 press
      MOVS  R4,  #8
      ANDS  R3,  R3,  R4
      CMP   R3,  #0
      BEQ   write_leds_SW3

      B     led_counter_Norm

led_counter_Norm:
    ADDS  R2,  R2,  #1 @ Implement normal led increase
    B     write_leds

led_counter_SW0:
      ADDS  R2,  R2,  #2 @ Implement fast-mode led increase(+2)
      B     write_leds

write_leds_SW2:
      MOVS  R5,  #0xAA
      STR   R5,  [R1, #0x14] @ Display 10101010 upon pressing SW2
      B     main_loop

write_leds_SW3:
      STR   R2,  [R1, #0x14] @ Hold the display to most recent value
      B     main_loop

write_leds:
      STR   R2,  [R1, #0x14] @ Display the current led pattern
      LDR   R3,  [R0, #0x10] @ Load idr data to check if SW1 is pressed
      MOVS  R4,  #2
      ANDS  R3,  R3,  R4
      CMP   R3,  #0
      BEQ   load_delay_fast @ Implement the corresponding delay depending on button
press
      B     load_delay_slow

load_delay_fast:
      LDR   R5,  =SHORT_DELAY_CNT @ Load corresponding delay data
      LDR   R6,  [R5]
      B     delay

load_delay_slow:
      LDR   R5,  =LONG_DELAY_CNT @ Load corresponding delay data
      LDR   R6,  [R5]
      B     delay

delay:
      subs  R6,  R6,  #1  @ Implement a do nothing look to act as a delay
      BNE   delay
      B     main_loop

@ LITERALS; DO NOT EDIT
      .align
RCC_BASE:                .word 0x40021000
AHBENR_GPIOAB:           .word 0b1100000000000000000
GPIOA_BASE:         .word 0x48000000
```

```
GPIOB_BASE:          .word 0x48000400
MODER_OUTPUT:              .word 0x5555

@ TODO: Add your own values for these delays
LONG_DELAY_CNT:    .word 1400000
SHORT_DELAY_CNT:   .word 600000
```