```
/*
 * 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

@ Main loop
main_loop:
  @ Read GPIOA IDR
  LDR R0, GPIOA_BASE
  LDR R3, [R0, #0x10]

  @ --- Step size: default 1, SW0 doubles step to 2 ---
  MOVS R4, #1
  MOVS R5, #0x01
  BL   debounce_button
  BNE  step_done
  MOVS R4, #2
step_done:

  @ --- SW2 priority: force 0xAA ---
  MOVS R5, #0x04
  BL   debounce_button
  BEQ  sw2_pressed

  @ --- SW3 priority: freeze current pattern ---
```

```
    MOVS R5, #0x08
    BL   debounce_button
    BEQ  sw3_pressed

    @ --- Normal counting (no SW2/SW3) ---
    @ SW1 selects short or long delay
    MOVS R5, #0x02
    BL   debounce_button
    BEQ  delay_short_normal

delay_long_normal:
    LDR  R6, LONG_DELAY_CNT
    B    delay_common_normal
delay_short_normal:
    LDR  R6, SHORT_DELAY_CNT
delay_common_normal:
delay_loop_normal:
    SUBS R6, R6, #1
    BNE  delay_loop_normal

    @ Update counter
    ADDS R2, R2, R4
    UXTB R2, R2
    B    write_leds


@ --- SW2 path: force 0xAA until release ---
sw2_pressed:
```

```
   MOVS R2, #0xAA
   MOVS R5, #0x02
   BL  debounce_button
   BEQ  delay_short_sw2
delay_long_sw2:
   LDR  R6, LONG_DELAY_CNT
   B   delay_common_sw2
delay_short_sw2:
   LDR  R6, SHORT_DELAY_CNT
delay_common_sw2:
delay_loop_sw2:
   SUBS R6, R6, #1
   BNE  delay_loop_sw2
   B   write_leds


@ --- SW3 path: freeze current pattern ---
sw3_pressed:
   MOVS R5, #0x02
   BL  debounce_button
   BEQ  delay_short_sw3
delay_long_sw3:
   LDR  R6, LONG_DELAY_CNT
   B   delay_common_sw3
delay_short_sw3:
   LDR  R6, SHORT_DELAY_CNT
delay_common_sw3:
delay_loop_sw3:
```

```
   SUBS R6, R6, #1
   BNE  delay_loop_sw3
   B    write_leds


@ Output LEDs
write_leds:
   STR R2, [R1, #0x14]
   B main_loop

@ Debounce subroutine
@ Input:  R0 = GPIOA_BASE
@         R5 = mask for switch
@ Output: returns with Z=0 if stable pressed, Z=1 otherwise

debounce_button:
   @ Quick delay (~10 ms, tune)
   LDR  R6, DEBOUNCE_CNT
db_delay_loop:
   SUBS R6, R6, #1
   BNE  db_delay_loop

   @ Re-read IDR
   LDR  R3, [R0, #0x10]
   ANDS R5, R3, R5
   BX   LR

@ 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

@ Delays
LONG_DELAY_CNT:  .word 700000
SHORT_DELAY_CNT:        .word 300000
DEBOUNCE_CNT:           .word 10000     @ ~10ms debounce
```