```
1                   .section .vectors, "ax"
2                   B       _start              // reset vector
3                   B       SERVICE_UND         // undefined instruction vector
4                   B       SERVICE_SVC         // software interrupt vector
5                   B       SERVICE_ABT_INST    // aborted prefetch vector
6                   B       SERVICE_ABT_DATA    // aborted data vector
7                   .word   0                   // unused vector
8                   B       SERVICE_IRQ         // IRQ interrupt vector
9                   B       SERVICE_FIQ         // FIQ interrupt vector
10                  .text
11                  .global _start
12  _start:
13  /* Set up stack pointers for IRQ and SVC processor modes */
14                  MOV     R1, #0b11010010     // interrupts masked, MODE = IRQ
15                  MSR     CPSR_c, R1          // change to IRQ mode
16                  LDR     SP, =0xFFFFFFFF - 3 // set IRQ stack to A9 onchip memory
17
18                  MOV     R1, #0b11010011     // interrupts masked, MODE = SVC
19                  MSR     CPSR, R1            // change to supervisor mode
20                  LDR     SP, =0x3FFFFFFF - 3 // set SVC stack to top of DDR3 memory
21
22                  BL      CONFIG_GIC          // configure the ARM generic interrupt
                    controller
23                  BL      CONFIG_TIMER        // configure the Interval Timer
24                  BL      CONFIG_KEYS         // configure the pushbutton KEYs port
25
26  /* Enable IRQ interrupts in the ARM processor */
27                  MOV      R0, #0b01010011    // IRQ unmasked, MODE = SVC
28                  MSR      CPSR_c, R0
29
30                  LDR     R5, =0xFF200000     // LEDR base address
31  LOOP:
32                  LDR     R3, COUNT           // global variable
33                  STR     R3, [R5]            // write to the LEDR lights
34                  B       LOOP
35
36  /* Configure the Interval Timer to create interrupts at 0.25 second intervals */
37  CONFIG_TIMER:   LDR     R0, =0xFF202000     // FPGA timer base address
38                  LDR     R1, =30784
39                  STR     R1, [R0, #8]        // Set lower bits of timer
40                  LDR     R1, =381
41                  STR     R1, [R0, #12]       // Set upper bits of timer
42                  MOV     R1, #0b0111         // Control register bits
43                  STR     R1, [R0, #4]        // Start timer, set auto-reload and enable
                    interrupts
44                  BX      LR
45
46  /* Configure the pushbutton KEYS to generate interrupts */
47  CONFIG_KEYS:
48                  LDR     R0, =0xFF200050     // KEY address
49                  MOV     R1, #0xF            // set interrupt mask bits
50                  STR     R1, [R0, #0x8]      // interrupt mask register (base + 8)
51                  BX      LR
52
53  /* Define the exception service routines */
54  SERVICE_IRQ:    PUSH    {R0-R7, LR}
55                  LDR     R4, =0xFFFEC100     // GIC CPU interface base address
56                  LDR     R5, [R4, #0x0C]     // read the ICCIAR in the CPU interface
57
58  FPGA_IRQ1_HANDLER:
59                  CMP     R5, #73             // check the interrupt ID
60                  BEQ     KEY_INTERRUPT
61                  CMP     R5, #72
62                  BEQ     CLK_INTERRUPT
63
64  UNEXPECTED:     B       UNEXPECTED          // if not recognized, stop here
65  KEY_INTERRUPT:  BL      KEY_ISR
66  CLK_INTERRUPT:  BL      TIMER_ISR
67
```

```
EXIT_IRQ:       STR     R5, [R4, #0x10]     // write to the End of Interrupt Register
(ICCEOIR)
                POP     {R0-R7, LR}
                SUBS    PC, LR, #4          // return from exception

/* Check if it has been 0.25 seconds and adds RUN to COUNT */
TIMER_ISR:      LDR     R0, =0xFF202000     // base address of FPGA timer
                LDR     R1, [R0]        // read edge capture register
                MOV     R2, #0
                STR     R2, [R0]            // clear the interrupt
                LDR     R0, =RUN            // Load RUN toggle
                LDR     R0, [R0]
                LDR     R1, =COUNT          // Load counter
                LDR     R2, [R1]
                ADD     R2, R0              // Increment counter by RUN
                STR     R2, [R1]            // Store incremented counter
END_TIMER_ISR:  BX      LR                  // Return

/* Check if a key has been pressed and toggles RUN */
KEY_ISR:        LDR     R0, =0xFF200050     // base address of pushbutton KEY port
                LDR     R1, [R0, #0xC]      // read edge capture register
                MOV     R2, #0xF
                STR     R2, [R0, #0xC]      // clear the interrupt
                LDR     R0, =RUN            // Address of RUN toggle

CHECK_KEYS:     MOV     R3, #0b1111
                ORRS    R3, R1              // Check for any KEY
                BEQ     END_KEY_ISR
                LDR     R1, [R0]            // Get RUN from memory
                EOR     R1, #1              // Toggle RUN
                STR     R1, [R0]            // Set RUN in memory

END_KEY_ISR:    BX      LR                  // Return

/* Global variables */
                .global COUNT
COUNT:          .word   0x0             // used by timer
                .global RUN                 // used by pushbutton KEYs
RUN:            .word   0x1                 // initial value to increment COUNT

                .end
```