```
 1 @ Project 2 - Part 3
 2 @ This program uses GPIO pins to turn on user LEDs and interface with a push
   button.
 3 @ The push button enables/disables a timer and the timer ISR controls the LED
   flash sequence.
 4 @ The current state of the LED flash sequence is stored in memory.
 5 @ Ryan Bentz - 12/2/17
 6
 7 .data
 8 .align 2
 9 LED_STATUS:      .word 0x00
10 STACK1:          .rept 1024
11                  .word 0x00
12                  .endr
13 STACK2:          .rept 1024
14                  .word 0x00
15                  .endr
16 .text
17 .global _start
18 .global INT_DIRECTOR
19 _start:
20
21 @ Define the Register Addressed, Offsets, and write values to control the LEDs
22 .equ    BUTTON_PIN, 0x40000000      @ constant for the Button pin/bit
23 .equ    LED_STATE0, 0x00000000      @ constant for ALL LEDs OFF
24 .equ    LED_STATE1, 0x00000001      @ constant for LED0 ON
25 .equ    LED_STATE2, 0x00000002      @ constant for LED1 ON
26 .equ    LED_STATE3, 0x00000003      @ constant for LED2 ON
27 .equ    LED_STATE4, 0x00000004      @ constant for LED3 ON
28 .equ    TIM_COUNT_VAL, 0xFFFF7FFE
29
30 @ Initialize the stack frames
31 @-----------------------------------------------------------------------------
32 LDR R13, =STACK1                    @ initialize stack one for supervisor mode
33 ADD R13, R13, #0x1000               @ point stack pointer to top of stack
34 CPS #0x12                           @ change to IRQ mode
35 LDR R13, =STACK2                    @ initialize stack for IRQ mode
36 ADD R13, R13, #0x1000               @ point stack pointer to top of stack
37 CPS #0x13                           @ change back to supervisor mode
38
39 @ Initialize the peripheral clocks
40 @-----------------------------------------------------------------------------
41 @ Enable the timer clock source in CMDLL at 32.768
42 LDR R0, =0x44E0050C     @ CM_DLL Timer 3 Register
43 LDR R1, [R0]            @ read the register value
44 MOV R2, #0x02           @ enable 32kHz clock
45 ORR R1, R1, R2          @ modify the current value
46 STR R1, [R0]            @ write new value to register
47
48 @ Enable GPIO 1 clock
49 LDR R0, =0x44E000AC     @ CM_PER GPIO1 Register
50 LDR R1, [R0]            @ Read the register value
51 MOV R2, #0x00000002     @ value to turn on the GPIO module
52 ORR R1, R1, R2          @ Combine new value and existing register value
53 STR R1, [R0]            @ Write the value to the register
54
55 @ Enable Timer 3 clock
56 LDR R0, =0x44E00084     @ CM_PER Timer 3 Register
57 LDR R1, [R0]            @ get the value of the register
58 MOV R2, #0x02
```

```
 59 ORR R1, R1, R2          @ modify the register
 60 STR R1, [R0]
 61
 62 @ Wait for periphals clocks to be ready for modification
 63 BL DELAY
 64
 65 @ Initialize the GPIO
 66 @ Set the LED pin state as low (OFF state)
 67 @ default values for GPIO pin states is OFF
 68
 69 @Configure the LED pins as output
 70 LDR R0, =0x4804C134     @ GPIO1_OE (output enable)
 71 LDR R1, [R0]            @ Read the register value
 72 @ Write zeroes to register bits to enable as output
 73 MOV R2, #0xFE1FFFFF     @ Value to enable the pin
 74 AND R1, R1, R2          @ Combine value to write new to register
 75 STR R1, [R0]            @ write new value to the register
 76
 77 @ initialize GPIO pin for falling edge detect
 78 LDR R0, =0x4804C14C     @ GPIO1_FALLING_DETECT
 79 LDR R1, [R0]            @ read the register value
 80 MOV R2, #BUTTON_PIN     @ word to program the register
 81 ORR R1, R1, R2          @ modify the register value
 82 STR R1, [R0]
 83
 84 @ Initialize the timer
 85 @ enable auto-reload for the timer overflow
 86 LDR R0, =0x48042038     @ load the address for TIM 3 CONTROL
 87 LDR R1, [R0]            @ read the register value
 88 MOV R2, #0x02           @ enable auto-reload
 89 ORR R1, R1, R2          @ modify the current value with new value
 90 STR R1, [R0]            @ write new value to the register
 91
 92 @ set the counter value
 93 LDR R0, =0x48042040     @ TIM3 Load Register
 94 LDR R1, =TIM_COUNT_VAL  @ load the counter value
 95 STR R1, [R0]            @ write new value to register
 96
 97 LDR R0, =0x4804203C     @ TIM3 Counter Register
 98 LDR R1, =TIM_COUNT_VAL
 99 STR R1, [R0]
100
101 @ Enable interrupts
102 @ initialize GPIO pin for external interrupt
103 LDR R0, =0x4804C034     @ GPIO1_IRQSTATUS_SET0
104 LDR R1, [R0]            @ read the register value
105 MOV R2, #BUTTON_PIN     @ load the word to program the register
106 ORR R1, R1, R2          @ modify the current value with new value
107 STR R1, [R0]            @ write new value to the register
108
109 @ enable timer IRQs for overflow
110 LDR R0, =0x4804202C     @ load the address for TIM3_IRQ_SET
111 LDR R1, [R0]            @ read the register value
112 MOV R2, #0x02           @ load the word to program the register
113 ORR R1, R1, R2          @ modify the current value with new value
114 STR R1, [R0]            @ write new value to the register
115
116 @ Initialize interrupt controller for Timer interrupts
117 @ DMTimer3 = # 69
118 LDR R0, =0x482000C8     @ INTC_MIR_CLEAR2
```

```
119 LDR R1, [R0]
120 MOV R2, #0x00000020     @ set bit 5 for interrupt 69
121 ORR R1, R1, R2          @ unmask Timer interrupt
122 STR R1, [R0]            @ write new value to the register
123
124 @ Initialize interrupt controller for GPIO pins
125 @ unmask interrupt by writing a 1 to MIR_CLEAR which sets the bit to 0
126 @ 0 = interrupt enabled, 1 = interrupt disabled
127 LDR R0, =0x482000E8     @ INTC_MIR_CLEAR3
128 LDR R1, [R0]            @ read the register value
129 MOV R2, #0x00000004     @ interrupt 98 = bit 3 in MIR3
130 ORR R1, R1, R2          @ unmask GPIO interrupt
131 STR R1, [R0]            @ write new value to the register
132
133
134 @ Initialize interrupts in the processor
135 MRS R3, CPSR    @ copy CPSR to R3
136 BIC R3, #0x80   @ clear bit 7
137 MSR CPSR_c, R3  @ write back to CPSR
138
139 MAIN_LOOP:
140     NOP
141     NOP
142     B MAIN_LOOP
143     B END
144
145
146 @-------------------------------------------------------------------------
147 @ INTERRUPT SERVICE ROUTINE:
148 @ This procedure is hooked into the interrupt vector table to supercede
149 @ processor IRQ requests. It checks if the button was the source of the
150 @ interrupt and runs a specific procedure if it is the source of the interrupt
151 INT_DIRECTOR:
152     STMFD SP!, {R0-R3, LR}      @ push registers on to the stack
153
154 @ check if GPIO was the interrupt source
155     LDR R0, =0x482000F8         @ INTC_PENDING_IRQ3
156     LDR R1, [R0]                @ read the pending interrupt register
157     MOV R2, #0x00000004         @ load test value to check if GPIO triggered
    interrupt
158     AND R1, R1, R2              @ test if GPIO was interrupt source
159     CMP R1, #0x00
160     BEQ TIMER_CHECK             @ if the result is zero, GPIO was not source
    of interrupt
161                                 @ if result is nonzero, GPIO triggered the
    interrupt
162                                 @ check to see if the button was the source of
    the GPIO interrupt
163     LDR R0, =0x4804C02C         @ GPIO1_IRQSTATUS_0
164     LDR R1, [R0]                @ read value of register
165     MOV R2, #BUTTON_PIN         @ load the value for the button pin
166     AND R1, R1, R2              @ make resultant word for comparison
167     CMP R1, #0x00               @ compare to see if result is nonzero.
168                                 @ Nonzero = GPIO flag was set
169     BNE BUTTON_SVC              @ If button was pressed go to button service
    procedure
170     B PASS_ON                   @ otherwise different GPIO caused interrupt
171     TIMER_CHECK:                @ ELSE, check if the timer caused the
    interrupt
172         LDR R0, =0x482000D8     @ INTC_PENDING_IRQ2
```

```
173         LDR R1, [R0]            @ read the pending interrupt register
174         MOV R2, #0x00000020     @ load test value to check if GPIO triggered
    interrupt
175         AND R1, R1, R2          @ test if GPIO was interrupt source
176         CMP R1, #0x00           @ if the result is nonzero, timer was the
    source of interrupt
177         BNE TIMER_SVC_RTN       @ otherwise timer was not the source
178     @ exit int director ISR
179     PASS_ON:
180         @ re-enable IRQ interrupts in the processor
181         MRS R3, CPSR    @ copy CPSR to R3
182         BIC R3, #0x80   @ clear bit 7
183         MSR CPSR_c, R3  @ write back to CPSR
184
185         LDMFD SP!, {R0-R3, LR}      @ restore register states
186         SUBS PC, LR, #4             @ return service to the system IRQ
187
188
189 @-------------------------------------------------------------------------
190 @ BUTTON SERVICE PROCEDURE
191 @ This procedure handles the specific button service requirements
192 @ It resets the GPIO IRQ flag and changes the LED flash status variable
193 @ IRQ #98
194 BUTTON_SVC:
195 @ push SPSR on stack
196     MRS R3, SPSR            @ copy the saved program status register
197     STMFD R13!, {R3}        @ push on to stack
198
199 @ mask lower priority interrupts
200     @ already masked
201
202 @ enable IRQ to allow higher priority IRQ interrupts
203     MRS R3, CPSR            @ copy CPSR to R3
204     ORR R3, #0x80           @ re-enable IRQs
205     MSR CPSR_c, R3          @ write back to lower bits of CPSR
206
207 @ reset the GPIO interrupt request
208     LDR R0, =0x4804C02C
209     MOV R2, #BUTTON_PIN
210     STR R2, [R0]
211
212 @ reset interrupt controller IRQ requests
213     LDR R0, =0x48200048     @ INTC_CONTROL
214     MOV R1, #0x01           @ value to reset IRQ generation
215     STR R1, [R0]
216
217 @ enable/disable the timer
218     LDR R0, =0x48042038     @ TIM3 Control Register
219     LDR R1, [R0]            @ read the register value
220     AND R2, R1, #0x00000001 @ mask all but the start bit
221     MOV R3, #0x00           @ load test value
222     CMP R2, R3              @ test to see if timer is off
223     MOVEQ R3, #0x03         @ put new value in R3
224     MOVNE R3, #0x02
225     STR R3, [R0]            @ update the register
226
227 @ set the counter value
228     LDR R0, =0x4804203C     @ TIM3 Counter Register
229     LDR R1, =TIM_COUNT_VAL
230     STR R1, [R0]
```

```
231
232 @ disable IRQ for exit code critical region
233     MRS R3, CPSR
234     ORR R3, R3, #0x80
235     MSR CPSR_c, R3
236
237 @ unmask lower priority interrupts
238 @ not necessary
239
240 @ restore SPSR
241     LDMFD R13!, {R3}
242     MSR SPSR_cf, R3
243     LDMFD R13!, {R0-R3, R14}
244     SUBS PC, LR, #4             @ return service to main
245
246
247 @------------------------------------------------------------------------
248 @ TIMER SERVICE ROUTINE
249 @ Handles the LED flashing sequence
250 TIMER_SVC_RTN:
251 @ Update the LEDs
252     LDR R0, =LED_STATUS         @ load the address for the state variable
253     LDR R2, [R0]                @ get the current state
254
255 @ IF LED STATE = 0 THEN ALL LEDs = OFF
256     MOV R3, #LED_STATE0         @ IF state == 0, ALL LEDs OFF, Turn on LED0
257     CMP R2, R3                  @ compare with state 1
258     MOVEQ R1, #0x00200000       @ if equal, turn on LED 0
259
260 @ IF LED STATE = 1 THEN LED 0 = ON
261     MOV R3, #LED_STATE1
262     CMP R2, R3                  @ compare with state 1
263     MOVEQ R1, #0x00600000       @ if equal, turn on LED 0, 1
264
265 @ IF LED STATE = 2 THEN LED 0 + 1 = ON
266     MOV R3, #LED_STATE2
267     CMP R2, R3                  @ compre with state 2
268     MOVEQ R1, #0x00E00000       @ if equal, turn on LED 0, 1, 2
269
270 @ IF LED STATE = 3 THEN LED 0 + 1 + 2 = ON
271     MOV R3, #LED_STATE3
272     CMP R2, R3                  @ compare with state 3
273     MOVEQ R1, #0x01E00000       @ if equal, turn on LED 0, 1, 2, 3
274
275 @ IF LED STATE = 4 THEN LED 0 + 1 + 2 + 3 = ON
276     MOV R3, #LED_STATE4
277     CMP R2, R3                  @ compare with state 4
278     MOVEQ R1, #0x00000000       @ if equal, turn off ALL LEDs
279
280 @ Update the GPIO register
281     LDR R0, =0x4804C13C         @ GPIO1_DATAOUT
282     STR R1, [R0]                @ update the GPIO register
283
284 @ Adjust the step variable
285     ADDS R2, #1                 @ increment the state variable
286     CMP R2, R3                  @ check if it has exceeded state 4
287     MOVHI R2, #LED_STATE0       @ reset to zero if greater than 4
288     LDR R0, =LED_STATUS         @ load the address for the state variable
289     STR R2, [R0]                @ update the step variable
290
```

```
291 @ Reset the Timer IRQ flags
292     LDR R0, =0x48042028         @ TIM3 IRQ Status Register
293     MOV R1, #0x2                @ mask to clear bit 2
294     STR R1, [R0]
295
296 @ Reset the interrupt controller IRQ flag
297     LDR R0, =0x48200048     @ INTC_CONTROL
298     MOV R1, #0x1            @ value to reset IRQ generation
299     STR R1, [R0]
300
301 @ re-enable IRQ interrupts in the processor
302     MRS R3, CPSR    @ copy CPSR to R3
303     BIC R3, #0x80   @ clear bit 7
304     MSR CPSR_c, R3  @ write back to CPSR
305
306     LDMFD SP!, {R0-R3, LR}      @ restore register states
307     SUBS PC, LR, #4             @ return service to main
308
309
310 @--------------------------------------------------------------------------
311 @ DELAY ROUTINE
312 @ Necessary to give some buffer after we turn on the peripheral clocks
313 @ before we start accessing registers
314 DELAY:
315 STMFD R13!, {R4, R14}   @ save the register states and link register location
316 LDR R4, =0x0022DCD5
317 D_LOOP:
318     NOP
319     SUBS R4, #1
320     BNE D_LOOP
321 LDMFD R13!, {R4, PC}
322
323 END:
324 .END
```