```
1 @ Ryan Bentz
2 @ 372 Project 2 - Part 2
3 @ This program interfaces with a New Haven LCD using the I2C
4 @ protocol in an interrupt based configuration.
5
6 .data
7 .align 2
8 @ define stack sizes
9 @-------------------------------------------------------------------
10 LED_STATUS:     .word 0x00
11 STACK1:         .rept 1024
12                 .word 0x00
13                 .endr
14 STACK2:         .rept 1024
15                 .word 0x00
16                 .endr
17
18 @ define word messages to send
19 @-------------------------------------------------------------------
20 .align 4
21 MESSAGE:        .ascii "RYAN BENTZ"
22 INIT_STREAM: .byte 0x38, 0x39, 0x14, 0x78, 0x5E, 0x6D, 0x6D, 0x0C, 0x01, 0x06,
   0x02
23 STREAM_OFFSET: .byte 0x00
24 INIT_INDEX: .byte 0x00
25 MESSAGE_INDEX: .byte 0x00
26
27 .text
28 .global _start
29 .global INT_DIRECTOR
30 _start:
31
32 @ Initialize the stack frames
33 @-------------------------------------------------------------------
34 LDR R13, =STACK1        @ initialize stack one for supervisor mode
35 ADD R13, R13, #0x1000   @ point stack pointer to top of stack
36 CPS #0x12              @ change to IRQ mode
37 LDR R13, =STACK2        @ initialize stack for IRQ mode
38 ADD R13, R13, #0x1000   @ point stack pointer to top of stack
39 CPS #0x13              @ change back to supervisor mode
40
41 @ Initialize the peripheral clocks
42 @-------------------------------------------------------------------
43 @ initialize the clock to I2C1
44 LDR R0, =0x44E00048     @ CMPER.I2C1_CLKCTRL
45 LDR R1, [R0]
46 MOV R2, #0x02          @ enable clock to module
47 ORR R1, R1, R2
48 STR R1, [R0]
49
50 @ Delay 1s and wait for peripheral clocks
51 @-------------------------------------------------------------------
52 BL DELAY
53
54 @ Initialize the I2C
55 @-------------------------------------------------------------------
56 @ initialize the GPIO pins for the I2C functions
57 @ set the SCL pin
58 LDR R0, =0x44E1095C               @ CONTROL.conf_spi0_cs0
```

```
 59 LDR R1, [R0]
 60 AND R1, R1, #0xFFFFFFF8             @ mask bits [2:0]
 61 ORR R1, R1, #0x02                   @ enable mode 2
 62 STR R1, [R0]
 63
 64 @ set the SDA pin
 65 LDR R0, =0x44E10958                 @ CONTROL.conf_spi0_d1
 66 LDR R1, [R0]
 67 AND R1, R1, #0xFFFFFFF8             @ mask bits [2:0]
 68 ORR R1, R1, #0x02                   @ enable mode 2
 69 STR R1, [R0]
 70
 71 @ set I2C ICLK prescaler
 72 LDR R0, =0x4802A0B0                 @ I2C_PSC: Clock prescaler register
 73 LDR R1, [R0]
 74 MOV R2, #0x3                        @ 48 MHz clock / 4 = 12 MHz clock
 75 ORR R1, R1, R2                      @ Prescaler val + 1
 76 STR R1, [R0]
 77
 78 @ set SCLL value
 79 LDR R0, =0x4802A0B4                 @ I2C_SCLL: Low time register
 80 LDR R1, [R0]
 81 MOV R2, #0x35
 82 ORR R1, R1, R2
 83 STR R1, [R0]
 84
 85 @ set SCLH value
 86 LDR R0, =0x4802A0B8                 @ I2C_SCLH: High time register
 87 LDR R1, [R0]
 88 MOV R2, #0x37
 89 ORR R1, R1, R2
 90 STR R1, [R0]
 91
 92 @ take the I2C module out of reset mode
 93 LDR R0, =0x4802A0A4                 @ I2C_CON: Control register
 94 LDR R1, [R0]
 95 MOV R2, #0x8000
 96 ORR R1, R1, R2
 97 STR R1, [R0]
 98
 99 @ configure I2C mode register without setting STT and STP
100 LDR R0, =0x4802A0A4                 @ I2C_CON: Control register
101 LDR R1, [R0]
102 LDR R2, =0xFFFF0000
103 AND R1, R1, R2
104 ORR R1, R1, #0x8600                 @ 7-bit address: 0x8600, 10-bit address:
    0x8700
105 STR R1, [R0]
106
107 @ configure the slave address
108 LDR R0, =0x4802A0AC                 @ I2C_SA: Slave address register
109 LDR R1, [R0]
110 AND R1, R1, #0x0000
111 ORR R1, R1, #0x3C
112 STR R1, [R0]
113
114 @ clear the FIFO
115 LDR R0, =0x4802A094     @ I2C_BUF Register
116 LDR R1, [R0]
```

```
117 LDR R2, =0x4040         @ clear TXFIFO and RXFIFO
118 ORR R1, R1, R2
119 STR R1, [R0]
120
121 @ Initialize the interrupt controller
122 @---------------------------------------------------------------------
123 @ initialize I2C interrupt
124 LDR R0, =0x482000C8 @ INTC_MIR_CLEAR2
125 LDR R1, [R0]            @ read the register value
126 MOV R2, #0x80           @ interrupt 71 = bit 7 in MIR2
127 ORR R1, R1, R2          @ unmask I2C interrupt
128 STR R1, [R0]            @ write new value to the register
129
130
131 @ Initialize interrupts in the processor
132 @---------------------------------------------------------------------
133 MRS R3, CPSR    @ copy CPSR to R3
134 BIC R3, #0x80   @ clear bit 7
135 MSR CPSR_c, R3  @ write back to CPSR
136
137
138 @ enable the I2C interrupt
139 LDR R0, =0x4802A02C         @ I2C_IRQENABLE_SET
140 LDR R1, [R0]
141 LDR R2, =0x106              @ Bus Free = bit 8, NACK = bit 1
142 ORR R1, R1, R2              @ enable BF and NACK interrupts
143 STR R1, [R0]
144
145 @ Flag ARDY to get the transfer started
146 LDR R0, =0x4802A024
147 LDR R1, [R0]
148 ORR R1, R1, #0x4
149 STR R1, [R0]
150
151 @ MAIN LOOP
152 @---------------------------------------------------------------------
153 MAIN_LOOP:
154     NOP             @ do nothing and wait for interrupt
155     B MAIN_LOOP
156     B END
157
158 @---------------------------------------------------------------------
159 @ INT Director handles the interrupt routines
160 INT_DIRECTOR:
161 STMFD SP!, {R0-R5, LR}  @ push registers on to the stack
162
163 @ check that I2C was the source of the interrupt
164 LDR     R0, =0x482000D8     @ INTC_PENDING_IRQ2
165 LDR     R1, [R0]            @ read the pending interrupt register
166 MOV     R2, #0x80           @ I2C interrupt = Bit 7
167 AND     R1, R1, R2
168 CMP     R1, #0x80
169 BEQ     INT_I2C
170
171 @ exit int director ISR
172 CLOSE_IRQ:
173 @ reset the interrupt controller IRQ flag
174 LDR R0, =0x48200048         @ INTC_CONTROL
175 MOV R1, #0x1                @ value to reset IRQ generation
```

```
176 STR R1, [R0]
177
178 @ re-enable IRQ interrupts in the processor
179 MRS R3, CPSR            @ copy CPSR to R3
180 BIC R3, #0x80           @ clear bit 7
181 MSR CPSR_c, R3          @ write back to CPSR
182
183 LDMFD SP!, {R0-R5, LR}  @ restore register states
184 SUBS PC, LR, #4         @ return service to the system IRQ
185
186 @---------------------------------------------------------------------
187 INT_I2C:
188 @ check if NACK response was recieved
189 LDR R0, =0x4802A028     @ I2C_IRQSTATUS Register
190 LDR R1, [R0]            @ get IRQ register value
191 AND R1, R1, #0x2        @ NACK = Bit 1
192 CMP R1, #0x00           @ If 0, it was something else
193 BNE INT_I2C_NACK
194
195 @ otherwise it was the BF or ARDY flag - do a transmission
196 @ clear the ARDY flag
197 LDR     R0, =0x4802A028         @ I2C_IRQSTATUS Register
198 LDR     R1, [R0]
199 ORR     R1, R1, #0x4
200 STR     R1, [R0]
201
202 @ check if we are doing initialization
203 LDR     R0, =INIT_INDEX
204 LDR     R1, =INIT_STREAM
205 MOV     R8, #0x00
206 LDRB    R2, [R0]
207 CMP     R2, #0xA                @ 9 bytes to send for initialization
208 BLE     I2C_TX_SETUP            @ branch to INIT if less than or equal to
209
210 @ check if we are sending message bytes
211 LDR     R0, =MESSAGE_INDEX
212 LDR     R1, =MESSAGE
213 MOV     R8, #0x40
214 LDRB    R2, [R0]
215 CMP     R2, #0x9
216 BLE     I2C_TX_SETUP
217
218 @ otherwise we are finished - mask the interrupts and exit
219 LDR     R0, =0x4802A030
220 LDR     R1, [R0]
221 ORR     R1, R1, #0x12           @ disable XRDY and NACK interrupts
222 STR     R1, [R0]
223
224 @ disable the I2C interrupt
225 LDR R0, =0x482000CC     @ INTC_MIR_SET2
226 LDR R1, [R0]            @ read the register value
227 MOV R2, #0x08           @ interrupt 71 = bit 7 in MIR2
228 ORR R1, R1, R2          @ mask I2C interrupt
229 STR R1, [R0]            @ write new value to the register
230
231 B       CLOSE_IRQ
232
233 @---------------------------------------------------------------------------
234 INT_I2C_NACK:
```

```
235 @ check if we are doing initialization
236 LDR     R0, =INIT_INDEX
237 LDR     R1, =INIT_STREAM
238 LDRB    R2, [R0]
239 CMP     R2, #0x9                @ 9 bytes to send for initialization
240 BLE     I2C_NACK_INIT           @ branch to INIT if less than or equal to
241
242 @ check if we are sending message bytes
243 LDR     R0, =MESSAGE_INDEX
244 LDR     R1, =MESSAGE
245 LDRB    R2, [R0]
246
247 I2C_NACK_INIT:
248 SUBS    R2, #2
249 STRB    R2, [R0]
250 B       I2C_TX_SETUP
251
252 @-------------------------------------------------------------------------
253 I2C_TX_SETUP:
254 @ R0 = OFFSET ADDRESS
255 @ R1 = STREAM ADDRESS
256 @ R8 = control byte
257 @ get the next byte to transmit
258 LDRB    R2, [R0]                @ get the offset value
259 ADD     R1, R1, R2              @ add the offset to the stream address
260 LDRB    R3, [R1]                @ get the next byte to send
261
262 @ load the control byte and the data byte into the FIFO
263 LDR     R1, =0x4802A09C         @ I2C_DATA Register
264 STRB    R8, [R1]                @ write the control byte 0x00 to the FIFO
265 STRB    R3, [R1]                @ write the data byte to the FIFO
266
267 @ increment the stream offset
268 LDRB    R1, [R0]                @ R0 = offset address
269 ADD     R1, R1, #0x1            @ increment the offset value
270 STRB    R1, [R0]                @ save the new value to memory
271
272 B       I2C_TX_BEGIN
273
274 @-------------------------------------------------------------------------
275 I2C_TX_BEGIN:
276 @ update the DCOUNT register
277 LDR R0, =0x4802A098     @ I2C_COUNT Register
278 MOV R1, #0x2
279 STR R1, [R0]
280
281 @ set module to master mode on every transfer
282 LDR     R0, =0x4802A0A4                 @ I2C_CON: Control register
283 LDR     R1, [R0]
284 ORR     R1, R1, #0x400                  @ Master mode bit 10
285 STR     R1, [R0]
286
287 @ configure the start/stop bits to begin transfer
288 LDR     R4, =0x4802A0A4     @ I2C_CON: Control Register
289 LDR     R5, [R4]
290 ORR     R5, R5, #0x03       @ Set the STT and STP bits to initiate transfer
291 STR     R5, [R4]            @ begin transmitting data by setting the
    start/stop bits
292
```

```
293 B   CLOSE_IRQ
294
295 @-------------------------------------------------------------------------
296 @ Delay Loop Subroutine
297 @ Handles the delay loop timing
298 DELAY:
299 STMFD R13!, {R0, R14}   @ save the register states and link register location
300 LDR R0, =0x0022DCD5
301 D_LOOP:
302     NOP
303     SUBS R0, #1
304     BNE D_LOOP
305 LDMFD R13!, {R4, PC}
306
307 END:
308 .END
```