THE UNIVERSITY OF TEXAS AT AUSTIN
EE 445L: EMBEDDED SYSTEMS DESIGN LAB

Lab 2 Prep

Joshua Bryant (jmb6357)   James Morris (jsm3288)

September 10, 2014

1. RxGetPt is located at 0x20000010 and RxPutPt is located at 0x2000000c.

2. a) **datapt** is an output parameter and is a call by reference. At the time of the call, **datapt** is stored on the stack.

   b) 0x00FC is stored at 0x00000A34. It is little endian. The memory address of what is being loaded is stored in R0 after the first LDR. The value at the previously loaded memory address is loaded into R0 after the second LDR is executed.

   c) The difference between **MOV** and **MOVS** is that the condition code flags are updated on the result of **MOVS** and the condition code flags are not updated on the result of **MOV**.

   d) The difference between **LDR** and **LDRB** is that **LDRB** zero extends to 32 bits on load.

   e) **LDR** is load and **STR** is store. Both are with immediate offset, pre-indexed immediate offset, or post-indexed immediate offset.

   f) **RET** is the return from subroutine command with the return address stored in a general purpose register specified by an operand for the opcode.

   g) A return parameter is passed by being stored in register r0. The parameter is stored in r0 at the time of the return.

3. The **StartCritical** and **EndCritical** are both added to **RxFifo_Init** to prevent interrupts from occurring during the assignment of the pointer variables for the FIFO to guarantee successful FIFO initialization. When **RxFifo_Init** returns, interrupts are either enabled or disabled, depending on if they were enabled or disabled before entering **RxFifo_Init**. The reason why is **EndCritical** restores the I bit to the value it had upon entering **RxFifo_Init**.

4.
```
0x000009C4 4601 MOV r1,r0
0x000009C6 481D LDR r0,[pc,#116] ;
@0x0A3C
0x000009C8 6800 LDR r0,[r0,#0x00]
0x000009CA 4A1B LDR r2,[pc,#108] ;
@0x0A38
0x000009CC 6812 LDR r2,[r2,#0x00]
0x000009CE 4290 CMP r0,r2
0x000009D0 D101 BNE 0x000009D6
0x000009D2 2000 MOVS r0,#0x00
0x000009D4 4770 BX lr
0x000009D6 4818 LDR r0,[pc,#96] ; @0x0A38
0x000009D8 6800 LDR r0,[r0,#0x00]
0x000009DA 7800 LDRB r0,[r0,#0x00]
0x000009DC 7008 STRB r0,[r1,#0x00]
0x000009DE 4816 LDR r0,[pc,#88] ; @0x0A38
0x000009E0 6800 LDR r0,[r0,#0x00]
0x000009E2 1C40 ADDS r0,r0,#1
0x000009E4 4A14 LDR r2,[pc,#80] ; @0x0A38
0x000009E6 6010 STR r0,[r2,#0x00]
0x000009E8 4610 MOV r0,r2
0x000009EA 6802 LDR r2,[r0,#0x00]
0x000009EC 4811 LDR r0,[pc,#68] ; @0x0A34
0x000009EE 3020 ADDS r0,r0,#0x20
0x000009F0 4282 CMP r2,r0
0x000009F2 D102 BNE 0x000009FA
0x000009F4 3820 SUBS r0,r0,#0x20
0x000009F6 4A10 LDR r2,[pc,#64] ; @0x0A38
0x000009F8 6010 STR r0,[r2,#0x00]
0x000009FA 2001 MOVS r0,#0x01
0x000009FC E7EA B 0x000009D4
0x00000A34 00FC LSLS r4,r7,#3

0x00000A36 2000 MOVS r0,#0x00
0x00000A38 0034 MOVS r4,r6
0x00000A3A 2000 MOVS r0,#0x00
0x00000A3C 0030 MOVS r0,r6
0x00000A3E 2000 MOVS r0,#0x00
```

Given an operating frequency of 80MHz, then the above highlighted code should take ~0.475 μs to execute.