Omar Marawi –Oam342

Keith Cyr –Kc32675

1) RxPutPt 0x2000000c RxGetPt 0x20000010 Data

2a) Char* datapt parameter is passed to rxfifo_get and the function will update the contents of datapt to the newest character. The pointer is an input but the contents within the ptr is part of the Output parameters. Call by reference. In r0, address =0x2000035

2b) A34- has the address of the array RXfifo. A38 has the address to the pointer Rxgetpt and A3C has the address of the pointer RXputPt. After the first LDR r0 has the address of RXputpt. After the second it has the pointer rxputpt. (Little or Big endian?)

2c) is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation.

2d) LDR will default to 32 bit loads LDRB specifies an 8 bits load from memory

2e) LDR reads memory, STR stores memory

2f) BX LR is return from sub routine if the LR is the previous pc counter at the location of the calling function. (to distinguish from Interrupts)

2g) passed by value, in r0

3) Because the fifo initialization is a critical section, the pointer to the fifo array is shared between the main program and the interrupt. The interrupt needs the fifo pointer to store new characters from the UART. If interrupts are not disabled we could overwrite new data before processing it. When the function returns, interrupts are set back to their previous state. We cannot definitively state if they are enabled or not.

4) **(HIGHLIGHT instructions executed in Fifo_Get - if FIFO is not full and pointer doesnt wrap. Add execution time of highlighted instructions and estimate time elapsed.) @ 80Mhz**

	INSTRUCTION	CYCLES
0x000009C4 4601	MOV r1,r0	1
0x000009C6 481D	LDR r0,[pc,#116] ; @0x0A3C	2^B(Barrier operation) = 2^(0.1..3)
0x000009C8 6800	LDR r0,[r0,#0x00]	2^B
0x000009CA 4A1B	LDR r2,[pc,#108] ; @0x0A38	2^B
0x000009CC 6812	LDR r2,[r2,#0x00]	2^B

```
0x000009CE 4290    CMP  r0,r2                          1
0x000009D0 D101    BNE  0x000009D6                     1+B= 1+(0,1,2,3)
0x000009D2 2000    MOVS r0,#0x00
0x000009D4 4770    BX   lr  (Last Instruction executed) 1+(0,1,2,3)
0x000009D6 4818    LDR  r0,[pc,#96]  ; @0x0A38  2^B
0x000009D8 6800    LDR  r0,[r0,#0x00]                2^B
0x000009DA 7800    LDRB r0,[r0,#0x00]                2^B
0x000009DC 7008    STRB r0,[r1,#0x00]               2^B
0x000009DE 4816    LDR  r0,[pc,#88]  ; @0x0A38       2^B
0x000009E0 6800    LDR  r0,[r0,#0x00]               2^B
0x000009E2 1C40    ADDS r0,r0,#1                     1
0x000009E4 4A14    LDR  r2,[pc,#80]  ; @0x0A38      2^B
0x000009E6 6010    STR  r0,[r2,#0x00]                2^B
0x000009E8 4610    MOV  r0,r2                        1
0x000009EA 6802    LDR  r2,[r0,#0x00]                2^B
0x000009EC 4811    LDR  r0,[pc,#68]  ; @0x0A34      2^B
0x000009EE 3020    ADDS r0,r0,#0x20                   1
0x000009F0 4282    CMP  r2,r0                        1
0x000009F2 D102    BNE  0x000009FA                   1 + B
0x000009F4 3820    SUBS r0,r0,#0x20
0x000009F6 4A10    LDR  r2,[pc,#64]  ; @0x0A38
0x000009F8 6010    STR  r0,[r2,#0x00]
0x000009FA 2001    MOVS r0,#0x01                  1
0x000009FC E7EA    B    0x000009D4    1 + B (Total ~ 50 instructions)(max=136)(min=25)


50 cycles approximately. At 80 Mhz that is 50/80M    =   5E1/8E7 = 0.625E-6 0.625 u
seconds
```