**Introduction**

**Procedure/Discussion**

**Analysis**

**Conclusion**

**References**

1. **JTAG Uart Data sheet**: <https://www-ug.eecg.utoronto.ca/desl/nios_devices_SoC/dev_jtaguart.html>
2. **ARMv7-M Architecture Reference Manual:** https://documentation- <https://documentation-service.arm.com/static/606dc36485368c4c2b1bf62f?token=>
   1. load instructions and derivatives(ldr) - A7.7.43
   2. store instructions and derivatives (str) - A7.7..161
   3. mov instruction (mov) - A7.7.76
   4. branch instructions - A7.7.12
   5. orr instruction - A7.7.89
   6. conditional execution mnemonics - A7.3
   7. Application Program Status Register (APSR) - A2.3.2
3. **GNU Assembler User Manual:** <https://www.ndsl.kaist.edu/ee209/references/gnu-assembler.pdf>
   1. .equ symbol, 7.21, page 45
   2. Labels, 5.1, page 33
4. **Loading constants in assembly for Arm Architecture:** <https://community.arm.com/arm-community-blogs/b/architectures-and-processors-blog/posts/how-to-load-constants-in-assembly-for-arm-architecture>

**Appendix**

.equ JTAG\_UART\_BASE\_ADDR, 0xFF201000

.equ JTAG\_UART\_DATA\_REG\_OFFSET, 0

.equ JTAG\_UART\_DATA\_VALID, (1<<7)

.equ JTAG\_UART\_CONTROL\_REG\_OFFSET, 4

.equ JTAG\_UART\_CONTROL\_WRITE\_IRQ\_PENDING, (1<<9)

.equ JTAG\_UART\_CONTROL\_READ\_IRQ\_PENDING, (1<<8)

.global \_start

\_start:

/\* r0 is global defined to be the base jtag uart register from now on \*/

ldr r0,=JTAG\_UART\_BASE\_ADDR

/\* INITIALIZATION. we put the jtag uart peripheral into a known and stable state

\* 1) disable both read and write interrupts by writing zero to bits 1, 0

\* 2) clear all pending interrupts by writing writing 1 to bits 9, 8

\*/

ldrb r1, =0

orr r1, r1, #(JTAG\_UART\_CONTROL\_WRITE\_IRQ\_PENDING |

JTAG\_UART\_CONTROL\_READ\_IRQ\_PENDING)

strb r1, [r0, #JTAG\_UART\_CONTROL\_REG\_OFFSET]

/\* POLL FOR CHARACTER INPUT. \*/

poll: ldrb r1, [r0, #(JTAG\_UART\_DATA\_REG\_OFFSET+1)]

ands r2, r1, #JTAG\_UART\_DATA\_VALID

beq poll

/\* extract the number of characters to read, r2 will hold the max loop count \*/

ldrh r2, [r0, #(JTAG\_UART\_DATA\_REG\_OFFSET+2)]

loop:

ldrh r1, [r0, #(JTAG\_UART\_DATA\_REG\_OFFSET)]

mov r3, #('a' - 'A') /\* bias value for upper case to lower case \*/

/\* check input is a letter (only letters can have capital or small versions) \*/

and r1, #0xFF /\* isolate the byte field which contains the letter \*/

cmp r1, #'A'

blt print

cmp r1, #'Z'

blt change\_case

mov r3, #('A' - 'a') /\* bias value for lower case to upper case \*/

cmp r1, #'a'

blt print

cmp r1, #'z'

bge decrement

change\_case:

add r1, r3

print:

str r1, [r0]

decrement:

adds r2, #-1 /\* decrement the loop counter \*/

bgt loop

nop

b . /\* stop processing here. \*/