Carleton University

Department of Systems and Computer Engineering

SYSC 3006 (Computer Organization) Fall 2020

Lab / Assignment 8 – Answers file

Student Name: ID#:

Part 1 – [1.5-mark/3]

1. [0.5-mark] Complete the LED fragment given in LEDSRC. The fragment includes the subroutine: void LED(uint LEDstate) to set the LED to the given LEDstate.
   Test your code to make sure it is working properly then enter your LEDSRC.txt final code here:

```
        EQU IObase, #0x80000000
        EQU ofsLED, #0x100
        EQU endOfStack, #0x800      ; initial SP value
        EQU breakpoint, #0xFFFFFFFF

        B main
IOaddrs DCD IObase

;void LED( uint LEDstate )    ; set the LED to the given LEDstate
LED    ; subroutine
        PUSH { R4, R14}
        LDR R4, [ IOaddrs ]         ; get IO base address
        STR R0, [R4, ofsLED]        ; set LED state = LEDstate
        POP { R4, R15 }

main
        MOV R13, endOfStack    ; initialize SP

        MOV R0, #1            ; set up LEDstate
        BL LED

         DCD breakpoint
```

   The LED subroutine will be used in subsequent fragments.

2. [0.5-mark] Complete the Switch fragment given in SwitchSRC.txt. The fragment includes the subroutines: int pollSwitchChange ( ) to poll Switch until switch changes state, return state after change. Also, the fragment includes the LED subroutine from above to turn off the led when switch rising edge is detected, then the program ends.
   Test your code to make sure it is working properly then enter your LEDSRC.txt final code here:

```
        EQU IObase, #0x80000000
        EQU ofsLED, #0x100
        EQU ofsSwitch, #0x200
        EQU endOfStack, #0x800      ; initial SP value
        EQU breakpoint, #0xFFFFFFFF


        B main
IOaddrs DCD IObase

; void LED( uint LEDstate )
; set the LED to the given LEDstate
LED    ; subroutine
        PUSH { R4, R14}
        LDR R4, [ IOaddrs ]            ; get IO base address
        STR R0, [R4, ofsLED]           ; set LED state = LEDstate
        POP { R4, R15 }

; int pollSwitchChange ( )
; poll switch until its value changes
; return the Switch state after it has changed
pollSwitchChange
        PUSH { R1,R4,R14 }
        LDR R4, [ IOaddrs ]             ; get IO base address
        LDR R1, [R4,ofsSwitch]          ; read initial Switch state
readSwitchAgain
        LDR R0, [ R4,ofsSwitch ]        ; read Switch state again
        CMP  R1,R0                      ; new state == initial state?
        BEQ readSwitchAgain             ; Yes! --> poll again
        POP { R1,R4,R15 }

main
        MOV R13, endOfStack            ; initialize SP

        MOV R0, #1    ; turn LED ON
        BL LED

        BL pollSwitchChange  ; wait for Switch to change

        MOV R0, #0           ; turn LED OFF
        BL LED

       DCD breakpoint
```

The pollSwitchChange subroutine will be used in in subsequent fragments.

3. [0.5-mark] Complete the HexSRC fragment given in HexSRC.txt. The fragment includes the subroutine: void HexON (displayValue) to turn ON Hex displays digits and display an initValue. Test your code to make sure it is working properly then enter your HexSRC.txt final code here:

```
        EQU IObase, #0x80000000
        EQU ofsLED, #0x100
        EQU ofsSwitch, #0x200
        EQU ofsHexCntl, #0x300
        EQU ofsHexData, #0x301
        EQU endOfStack, #0x800       ; initial SP value
        EQU breakpoint, #0xFFFFFFFF

        B main
IOaddrs DCD IObase


; void HexON ( uint initValue )
HexON
        PUSH { R1, R4, R14 }
        LDR R4, [IOaddrs ]             ; get IO base address
        MOV R1, #3                     ; get control value to turn on hex display digits (0,1,2,3)
        STR R1, [R4, ofsHexCntl]       ; (insert complete instruction) turn on hex display digits
        STR R0, [R4, ofsHexData]       ; (insert complete instruction) display initValue
        POP { R1, R4, R15}


main
        MOV R13, endOfStack    ; initialize SP

        MOV R0, #0x4F  ; turn on Hex display and display a value. be sure to try different values
        BL HexON

       DCD breakpoint
```

The HexON subroutine will be used in part 2.

Part 2 – [1.5-mark/3]

1. [1.5-mark] Complete the CountSRC fragment given in CountSRC.txt. The fragment includes the subroutines:

    1- void HexOUT (displayValue) to display a count value

    2- bcd2toBCD (uint Value) to convert a count value to 2-digit BCD encoding (assumes Value < 100). Recall BCD encoding from the Information Encoding slides.

    To compete your project, add all subroutines from part 1 to CountSRC fragment and test it to make sure it is working properly as follow:

    The count on the seven segments display should get incremented at each Switch state change from 0 to 1. When the count reaches 20, the LED will go on and the program ends, Enter your CountSRC.txt final working code here:

```
          EQU IObase, #0x80000000
          EQU ofsLED, #0x100
          EQU ofsSwitch, #0x200
          EQU ofsHexCntl, #0x300
          EQU ofsHexData, #0x301
          EQU endOfStack, #0x800       ; initial SP value
          EQU breakpoint, #0xFFFFFFFF
          EQU switchOFF, #0
          EQU switchON, #1

          B main
IOaddrs DCD IObase


;void LED( uint LEDstate )  ; set the LED to the given LEDstate
LED    ; subroutine
         PUSH { R4, R14}
         LDR R4, [ IOaddrs ]        ; get IO base address
         STR R0, [R4, ofsLED]       ; set LED state = LEDstate
         POP { R4, R15 }

; int pollSwitchChange ( )
; poll switch until its value changes
; return the Switch state after it has changed
pollSwitchChange
         PUSH { R1,R4,R14 }
         LDR R4, [ IOaddrs ]           ; get IO base address
         LDR R1, [R4,ofsSwitch]        ; read initial Switch state
readSwitchAgain
         LDR R0, [ R4,ofsSwitch ]        ; read Switch state again
         CMP  R1,R0                  ; new state == initial state?
         BEQ readSwitchAgain        ; Yes! --> poll again
         POP { R1,R4,R15 }

; void HexON ( uint initValue )
HexON
         PUSH { R1, R4, R14 }
         LDR R4, [IOaddrs ]           ; get IO base address
         MOV R1, #3  ; get control value to turn on hex display digits(0,1,2,3)
         STR R1, [R4, ofsHexCntl]  ; (insert complete instruction) turn
                                   ; on hex display digits
         STR R0, [R4, ofsHexData]  ; (insert complete instruction) display initValue
         POP { R1, R4, R15}

; void HexOUT ( uint displayValue )
HexOUT
         PUSH { R4, R14 }
         LDR R4, [ IOaddrs ]            ; get IO base address
         STR R0, [R4, ofsHexData]; (insert complete instruction) display displayValue
         POP { R4, R15 }


; bcd2 toBCD ( uint Value )
; assumes Value < 100(dec)
; returns BCD representation of Value in lower byte of return value
toBCD
         PUSH { R1,R2,R14 }
         DIV R1, R0, #0xA      ; generate BCD digits
         AND R2, R1, #0xF      ; isolate most signif digit
         LSL R0, R2, #0x4      ; put most signif digit in return value
         LSR R2, R1, #16       ; isolate least signif digit
         ADD R0, R0, R2        ; (insert complete instruction) put least signif digit
                               ; in return value
         POP { R1,R2,R15 }

main
         MOV R13, endOfStack    ; initialize SP

         MOV R1, #0x0    ; R1 = Count

         BL HexON              ; display initial value = 00

countEdge      ; this loop counts rising edges of the switch, stops when Count = 20

         BL  pollSwitchChange   ; wait for switch to change

         CMP R0, switchON      ; new state == ON (therefore a rising edge)?
         BLNE pollSwitchChange   ; No --> wait for rising edge
                                 ; or BNE countEdge

         ADD R1, R1, #1    ; increment Count

         MOV R0, R1       ; display current Count
         BL toBCD
         BL HexOUT

         CMP R1, #20       ; done counting?
         BNE  countEdge    ; No --> get next switch change

         MOV R0, switchON     ; (insert complete instruction)
         BL LED  ; Turn LED ON

        DCD breakpoint
```

Must be submitted on cuLearn, locate (Assignment 8 submission) and follow instructions. Submission exact deadline (date and time) is displayed clearly within the Assignment 8 submission on cuLearn.

Note: If you have any question please contact your respective group TA (see TA / group information posted on cuLearn) or use Discord class server.

Good Luck