Lab2 Report

```
;*---------------------------------------------------------------------------
;* Name:    Lab_2_program.s
;* Purpose: This code template is for Lab 2
;* Author: Eric Praetzel and Rasoul Keshavarzi
;*---------------------------------------------------------------------------*/
                THUMB           ; Declare THUMB instruction set
        AREA            My_code, CODE, READONLY      ;
        EXPORT          __MAIN                  ; Label __MAIN is used externally q
        ENTRY
__MAIN
; The following operations can be done in simpler methods. They are done in this
; way to practice different memory addressing methods.
; MOV moves into the lower word (16 bits) and clears the upper word
; MOVT moves into the upper word
; show several ways to create an address using a fixed offset and register as offset
;   and several examples are used below
; NOTE MOV can move ANY 16-bit, and only SOME >16-bit, constants into a register
; BNE and BEQ can be used to branch on the last operation being Not Equal or EQual to zero
;
        MOV             R2, #0xC000             ; move 0xC000 into R2
        MOV             R4, #0x0                ; init R4 register to 0 to build address
        MOVT            R4, #0x2009             ; assign 0x20090000 into R4
        ADD             R4, R4, R2              ; add 0xC000 to R4 to get 0x2009C000

        MOV             R3, #0x0000007C         ; move initial value for port P2 into R3
        STR             R3, [R4, #0x40]  ; Turn off five LEDs on port 2
```

```
        MOV             R3, #0xB0000000         ; move initial value for port P1 into R3
        STR             R3, [R4, #0x20]  ; Turn off three LEDs on Port 1 using an offset

ResetLUT
                LDR     R5, =InputLUT           ; assign R5 to the address at label LUT

NextChar
    LDRB    R0, [R5]                ; Read a character to convert to Morse
    ADD     R5, #1          ; point to next value for number of delays, jump by 1 byte
                TEQ     R0, #0          ; If we hit 0 (null at end of the string) then reset to the start of
lookup table
                BNE             ProcessChar             ; If we have a character process it

                MOV             R0, #4                  ; delay 4 extra spaces (7 total) between
words
                BL              DELAY
                BEQ     ResetLUT

ProcessChar     BL          CHAR2MORSE          ; convert ASCII to Morse pattern in R1

;       This is a different way to read the bits in the Morse Code LUT than is in the lab manual.
;       Choose whichever one you like.
;
;       First - loop until we have a 1 bit to send  (no code provided)
;
;       This is confusing as we're shifting a 32-bit value left, but the data is ONLY in the lowest 16 bits,
so test starting at bit 15 for 1 or 0
;       Then loop thru all of the data bits:
;
```

```
RemoveZero
        MOV     R6, #0x10000    ; Init R6 with the value for the bit, 17th, which we wish to test
        LSL     R1, R1, #1      ; shift R1 left by 1, store in R1
        ANDS    R7, R1, R6      ; R7 gets R1 AND R6, Zero bit gets set telling us if the bit is 0 or 1
        BEQ     RemoveZero      ; branch somewhere it's zero
        LSR     R1, R1, #1      ; shift the R1 to the righy by one bit, so we reset to R1 to 17th when the program come to BLINK
        BNE     BLINK           ; When the situation is R7 is not rqual to zero, go to BLINK

BLINK
    MOV     R6, 0x10000   ; Init R6 with the value for the bit, 17th, which we wish to test
        LSL     R1, R1, #1      ; shift R1 left by 1, store in R1
        ANDS    R7, R1, R6      ; R7 gets R1 AND R6, Zero bit gets set telling us if the bit is 0 or 1
        BEQ     LIGHT_OFF       ; when R7 gets 0 bit, means in that bit the light is off, go to the light-off subroutine.
        B       LIGHT_ON        ; otherwise, light is on

LIGHT_OFF
    MOV     R0, #1      ; Init R0 is euqal to 1, the light off when it still in the one letter.
        BL      LED_OFF     ;  go to turn off subroutine
        B       TEST_END    ; after the turnoff, we need to test if we fiinshed the letter.
JUMP    BL      DELAY       ; after we test the end, if it is still in the same letter, go to 500ms to wait
    B       BLINK       ; after that , go back to do BLINK again

LIGHT_ON
    MOV     R0, #1      ;  Init R0 is euqal to 1, the light off when it still in the one letter.
        BL      LED_ON      ;  go to turn on subroutine
```

```
                BL      DELAY     ; after the turnon, we need to test if we fiinshed the letter.
                B       BLINK     ; after that , go back to do BLINK again

TEST_END
    MOV     R6, #0x8000   ; Init R6 with the value for the bit, 16th, which we wish to test
                ANDS    R7, R1, R6    ;R7 gets R1 AND R6, Zero bit gets set telling us if the bit is 0 or 1
                BEQ     CHAR_DELAY    ; if R7 is 0,  then it is finished the letter and go to the next letter
                BNE     JUMP          ; otherwise, go to lelay or BLINk

CHAR_DELAY
    MOV     R0, #3        ; Init R0 is euqal to 3
                BL      DELAY         ; wait for 3 500ms
                B       NextChar      ; go to the next char




; Subroutines
;
;                       convert ASCII character to Morse pattern
;                       pass ASCII character in R0, output in R1
;                       index into MorseLuT must be by steps of 2 bytes
CHAR2MORSE   STMFD              R13!,{R14}        ; push Link Register (return address) on stack
                SUB         R0, R0, #0x00000041
                            ADD         R0, R0, R0
        LDR         R11, =MorseLUT
        LDRH        R1, [R11, R0]
```

```
                LDMFD           R13!,{R15}      ; restore LR to R15 the Program Counter to return


; Turn the LED on, but deal with the stack in a simpler way
; NOTE: This method of returning from subroutine (BX  LR) does NOT work if subroutines are nested!!
;
LED_ON          push        {r3-r4}         ; preserve R3 and R4 on the R13 stack
                mov     R3, #0xA0000000   ;move initial value for port P1 into R3
                STR     R3, [R4, #0x20]   ;turn on three LEDs on Port 1 using an offset
                pop         {r3-r4}
                BX              LR          ; branch to the address in the Link Register.  Ie return to the caller


; Turn the LED off, but deal with the stack in the proper way
; the Link register gets pushed onto the stack so that subroutines can be nested
;
LED_OFF         STMFD       R13!,{R3, R14}   ; push R3 and Link Register (return address) on stack
                mov     R3, #0xB0000000   ;move initial value for port P1 into R3
                STR     R3, [R4, #0x20]   ;turn off three LEDs on Port 1 using an offset
                LDMFD           R13!,{R3, R15}   ; restore R3 and LR to R15 the Program Counter to return


;       Delay 500ms * R0 times
;       Use the delay loop from Lab-1 but loop R0 times around
;
DELAY           STMFD       R13!,{R2, R14}   ; 500ms delay

MultipleDelay
```

```
        MOV     R9, #0xB0000   ; ini R9 to 000B0000

Loop    SUBS    R9, #0x0001    ; min 1 in R9 every time as a counter.
                BNE     Loop
        SUBS    R0, #0x0001    ; how many 1 in R0, it means how many 500ms delay in the light
        BEQ     exitDelay      ; go to delay
        BNE     MultipleDelay          ; loop the delay
exitDelay       LDMFD           R13!,{R2, R15}

;
; Data used in the program
; DCB is Define Constant Byte size
; DCW is Define Constant Word (16-bit) size
; EQU is EQUate or assign a value.  This takes no memory but instead of typing the same address in many places one can just use an EQU
;
        ALIGN                   ; make sure things fall on word addresses

; One way to provide a data to convert to Morse code is to use a string in memory.
; Simply read bytes of the string until the NULL or "0" is hit.  This makes it very easy to loop until done.
;
InputLUT    DCB         "ZZZZZ", 0      ; strings must be stored, and read, as BYTES

        ALIGN                   ; make sure things fall on word addresses
MorseLUT
        DCW     0x17, 0x1D5, 0x75D, 0x75        ; A, B, C, D        Morse tsble when er can refer to use
        DCW     0x1, 0x15D, 0x1DD, 0x55         ; E, F, G, H
```

```
                DCW     0x5, 0x1777, 0x1D7, 0x175       ; I, J, K, L
                DCW     0x77, 0x1D, 0x777, 0x5DD        ; M, N, O, P
                DCW     0x1DD7, 0x5D, 0x15, 0x7         ; Q, R, S, T
                DCW     0x57, 0x157, 0x177, 0x757       ; U, V, W, X
                DCW     0x1D77, 0x775                   ; Y, Z

; One can also define an address using the EQUate directive
;
LED_PORT_ADR    EQU     0x2009c000      ; Base address of the memory that controls I/O like LEDs

                END
```

Lab-2 Submission form

201 ☐	202 ☑	203 ☐	Demo date: Oct. 21 2016
204 ☐	205 ☐	206 ☐	

Submission Statement: We (I) are (am) submitting this report for grading in ECE 222. We (I) certify that this report (including any code, descriptions, flowcharts, etc., that are part of the submission) were written by us (me) and have received no prior academic credit at this university or any other institution. **The penalty for copying or plagiarism will be a grade of zero (0).**

Member 1	Member 2
Name: Zhang Zaoli	Name: ~~Fan~~ Gao Tiancheng
UW-ID (**NOT** student #) z376zhan	UW-ID (**NOT** student #) T35Gao
Signature:	Signature:

Note: Reports submitted without a signed submission statement will receive a grade of zero (0).

		Weight	Grade	Comment
Part-I	Pre-lab	0		
Part-II	Lab completion	40	40	
Lab-demo	Questions	40	40	
Part-III	Code quality	10	10	
Lab report	Code comments	10	10	
Penalty for using flash memory for code development		-20		
	Total	100		

Marking TA: Arvind M. Kaushik