

;\*----------------------------------------------------------------------------

;\* 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

MOV R2, #0xC000

MOV R3, #0xB0000000

MOV R4, #0x0

MOVT R4, #0x2009

ADD R4, R4, R2 ; 0x2009C000 - the base address for dealing with the ports

STR R3, [r4, #0x20] ; Turn off the three LEDs on port 1

MOV R3, #0x0000007C

STR R3, [R4, #0x40] ; Turn off five LEDs on port 2

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)

;use R2 as a counter for thr number of iterations. Register can only hold 16 bits, thus set the init value to 16

MOV R6, #0x8000

MOV R2 , #16

;keep left shifting the char until we get rid of all the leading 0s

RemoveLeadingZeroes

LSL R1, R1, #1

SUB R2, R2, #1

ANDS R7, R1, R6

BEQ RemoveLeadingZeroes

NextBit

;depending on the value of R1 we need to turnt the led on/off, thus do a logical AND and on each bit

ANDS R7, R1, R6

BLEQ LED\_OFF ; turn off the led

BLNE LED\_ON ; turn led on

LSL R1, R1, #1 ; done processesing the current char, now look at the next one

MOV R0 , #1 ; used for delay (.5ms each val in a char)

SUB R2, R2, #1 ; update our counter

BL DELAY

CMP R2, #0x0

BNE NextBit

MOV R0 , #3 ; 3ms delay betweene each char

BL LED\_OFF

BL DELAY

B NextChar ; This is the end of the main program

; Subroutines

CHAR2MORSE STMFD R13!,{R10, R11, R14}

; take the current char and subtract. once that is done we can multiply it by 2 and lookup the corresponding morsecode in

;the MorseLUT table

SUB R0, R0, #0x00000041

MOV R10, #0x00000002

MUL R0, R10, R0

LDR R11, =MorseLUT

LDRH R1, [R11, R0]

LDMFD R13!,{R10, R11, R15} ; restore LR to R15 the Program Counter to return

LED\_ON push {r3-r4} ; preserve R3 and R4 on the R13 stack

MOV R3, #0xA0000000 ;

STR R3, [R4, #0x20]

pop {r3-r4}

BX LR ; branch to the address in the Link Register. Ie return to the caller

LED\_OFF STMFD R13!,{R3, R14} ; push R3 and Link Register (return address) on stack

MOV R3, #0xB0000000 ;

STR R3, [R4, #0x20]

LDMFD R13!,{R3, R15} ; restore R3 and LR to R15 the Program Counter to return

DELAY STMFD R13!,{R9, R14}

MultipleDelay TEQ R0, #0 ; test R0 to see if it's 0 - set Zero flag so you can use BEQ, BNE

MOV R9, #0xFFFF

MOVT R9, #0x0005

MUL R0, R0, R9

Loop

SUBS R0, #1 ; Decrement r0 and set the N,Z,C status bits

BNE Loop

exitDelay LDMFD R13!,{R9, R15}

InputLUT DCB "AI", 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

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