; ECE-222 Lab ... Winter 2013 term

; Lab 3 sample code

THUMB ; Thumb instruction set

AREA My\_code, CODE, READONLY

EXPORT \_\_MAIN

ENTRY

\_\_MAIN

; The following lines are similar to Lab-1 but use a defined address to make it easier.

; They just turn off all LEDs

LDR R9,=FIO2PIN ;get the address of fio2pin and load it into R9

LDR R6,[R9] ;get the contents of R9 and store it into R6, going to be high or low(read the input of button)

MOV R11, #0xABCD ;Init the random number generator with a non-zero number

Start

BL RandomNum ;get the random number to R11

LDR R6,[R9] ; check the value of R6, i.e. the INT0 button

ANDS R6, #0x400 ; check the 14th bit, which is the on-off bit

BNE Start ; loop back to Start if the button is not pressed

LDR R10, =LED\_BASE\_ADR ; R10 is a permenant pointer to the base address for the LEDs, offset of 0x20 and 0x40 for the ports

MOV R3, #0xB0000000 ; Turn off three LEDs on port 1

STR R3, [r10, #0x20]

MOV R3, #0x0000007C

STR R3, [R10, #0x40] ; Turn off five LEDs on port 2

; This line is very important in your main program

; Initializes R11 to a 16-bit non-zero value and NOTHING else can write to R11 !!

;stepB

;LSR R2,R11,#1 ;Shift R11 to the right be 1 and load this random value into R2,

MOV R0, R11 ;Move the value aquired in R11 to another temporary register

LSR R0, #1 ;shift the random sequence right by 1, so that we can get the bit at position 1

BFC R0, #5, #27 ;clear the bits from bit 5 to the next 27 bits, since we only want the bits from position 1 to 5

;MOV R0, #5;

subs R0, #2 ; substract that value by 2

BLT RandomNum ; if that value is less than zero, meaning the initial value is less thatn two, create another ;Random number

adds R0,#2 ;passed, get the value back

subs R0, #10 ;substract the value by 10

BGT RandomNum ; if it's greater thatn 10, create another random number

adds R0,#10 ;passed, get the value back

MOV R8, #0 ; initialize the reflex conter to 0

MOV R5, #0x2710 ; 10000, conversion from 0.1 milisecond to second

MUL R0, R0, R5 ; multiply the random number with the conversion value to second

BL DELAY ; start the delay befor the LED turns on

BL LED\_ON ; turn on the indicatior LED

BL Polling ; count the button press delay

BL LED\_OFF ; turn off the LED

B DISPLAY\_NUM ; display the resulting value

;B loop

;

; Display the number in R3 onto the 8 LEDs

DISPLAY\_NUM STMFD R13!,{R1, R2, R3, R14}

MOV R4, #4; ; there are 4 sets to be displayed

MOV R1, R8 ; move the original reflex counter value to a temporary register so that it can be altered

; Usefullcommaands: RBIT (reverse bits), BFC (bit field clear), LSR & LSL to shift bits left and right, ORR & AND and EOR for bitwise operations

DISPLAY\_8

PORT2

BFI R2, R1, #25, #5 ; insert to R2 from position 25 onwards the 5 bit from R1 starting at position 0. 25 is the coresponding position before the reverse (R2 will be the register for port 2 LEDs)

RBIT R2, R2 ; reverse the bits

EOR R2, #0xFF ; flip all the bits from position 0 to 7

STR R2, [R10, #0x40] ; store it into the LED's address

PORT1

LSR R1, #5; ; shift the counter by 5, since we finished moving all the bits for Port 2

BFI R3, R1, #0, #1 ; insert into R3 position 0 the 1 bit from R1 at position 0. R3 will be the register for port 1 LEDs

LSR R1, #1; ; shift R1 right by 1, since we are done inserting the value for LED at port P1.31.

BFI R3, R1, #2, #2 ; insert into R3 position 2 the 2 bits from R1 at position 0. There is a 1 bit gap since the other two LEDs are on P1.29 and P1. 28

RBIT R3, R3 ; reverse the bits

EOR R3, #0xF0000000 ; flip the bits at position 28 to 31

STR R3, [R10, #0x20] ; store it into the LED's address

MOV R0, #2 ; make a 2s delay between display

MUL R0, R5 ; multiply with the 0.1ms to 1s converter

BL DELAY ; start the delay

LSR R1, #2 ; shift the reflex counter by 2 after inserting the P1.28 and P1.29 value

SUBS R4, #1 ; check the 4-display counter

BNE DISPLAY\_8 ; if stl has bits to display, loop back.

MOV R0, #5 ; start a 5s delay counter

MUL R0, R5 ; multiply with the converter

BL DELAY ; start the delay

B DISPLAY\_NUM ; re-display the reflex counter

LDMFD R13!,{R1, R2, R3, R15}

Polling

STMFD R13!, {R1, R2, R3, R14}

count

MOV R7, #0x07D ; Resetting counter to #0x7D, the 0.1 ms counter

ADD R8, #1 ; add the reflex counter by 1

xloop

SUBS R7, #1 ; Decrement r1, teh 0.1ms counter

BNE xloop ;

LDR R6,[R9] ; check whether the INT0 button has been pressed

ANDS R6, #0x400

BNE count ; redo the reflex counter

LDMFD R13!,{R1, R2, R3, R15}

;

; R11 holds a 16-bit random number via a pseudo-random sequence as per the Linear feedback shift register (Fibonacci) on WikiPedia

; R11 holds a non-zero 16-bit number. If a zero is fed in the pseudo-random sequence will stay stuck at 0

; Take as many bits of R11 as you need. If you take the lowest 4 bits then you get a number between 1 and 15.

; If you take bits 5..1 you'll get a number between 0 and 15 (assuming you right shift by 1 bit).

;

; R11 MUST be initialized to a non-zero 16-bit value at the start of the program OR ELSE!

; R11 can be read anywhere in the code but must only be written to by this subroutine

RandomNum STMFD R13!,{R1, R2, R3, R14}

AND R1, R11, #0x8000

AND R2, R11, #0x2000

LSL R2, #2

EOR R3, R1, R2

AND R1, R11, #0x1000

LSL R1, #3

EOR R3, R3, R1

AND R1, R11, #0x0400

LSL R1, #5

EOR R3, R3, R1 ; the new bit to go into the LSB is present

LSR R3, #15

LSL R11, #1

ORR R11, R11, R3

LDMFD R13!,{R1, R2, R3, R15}

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, [R10, #0x20] ; Turn off 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

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, [R10, #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 0.1ms (100us) \* R0 times

; aim for better than 10% accuracy

DELAY STMFD R13!,{R2, R14}

;

; code to generate a delay of 0.1mS \* R0 times

;

Mult

SUBS R0, #1;

MOV R7, #0x07D ; Resetting counter to #0x7FFFF

dloop

SUBS R7, #1 ; Decrement r7, the 0.1ms counter

BNE dloop ;

CMP R0, #0;

BNE Mult;

exitDelay LDMFD R13!,{R2, R15}

LED\_BASE\_ADR EQU 0x2009c000 ; Base address of the memory that controls the LEDs

PINSEL3 EQU 0x4002c00c ; Address of Pin Select Register 3 for P1[31:16]

PINSEL4 EQU 0x4002c010 ; Address of Pin Select Register 4 for P2[15:0]

FIO1PIN EQU 0x2009c034

FIO2PIN EQU 0x2009c054

; Usefull GPIO Registers

; FIODIR - register to set individual pins as input or output

; FIOPIN - register to read and write pins

; FIOSET - register to set I/O pins to 1 by writing a 1

; FIOCLR - register to clr I/O pins to 0 by writing a 1

ALIGN

END

Lab Report Answer:

1. 8-bits: 0.0255 second  
   16-bits: 6.5535 seconds  
   24-bits: 1677.7215 seconds = 27.962025 minutes  
   32-bits: 429496.7295 seconds = 119.3046470833 hours
2. Since 8-bits can only records to 0.0255 seconds, that’s a little too fast for human reaction time. Using 16-bits would be the best for this task because it shows enough significant digit to show the human reaction time, since on average human reaction time cannot go more than 6.5 seconds.