ECE243 Lab5
Parts 1-3
Nathan Jones

Part 1:

```
.global _start
_start:

//Store important things in memory to registers
MOV R0, #HEX_ADDR
LDR R0, [R0]
MOV R1, #KEY_ADDR
LDR R1, [R1]
MOV R2, #HEX_BITS
MOV R3, #0

MOV R6, #0
MOV R9, #0

//Clear R2
LDRB R9, [R2, #1]
STR R9, [R0]

MAIN:

        //R6 Holds value of which key is pressed
        LDR R7, [R1]
        MOV R6, R7

        //Wait for a key Press
        CMP R7, #0
        BEQ MAIN

        //Wait for the key to be released
        WAIT:
                LDR R7, [R1]
                CMP R7, #0
                BNE WAIT

        //See if KEY0 was pressed
        CMP R6, #1
        BEQ KEY_ZERO
```

```
        //See if KEY1 was pressed
        CMP R6, #2
        BEQ KEY_ONE

        //See if KEY2 was pressed
        CMP R6, #4
        BEQ KEY_TWO

        //See if KEY3 was pressed
        CMP R6, #8
        BEQ KEY_THREE

        B MAIN

//Reset to Zero
KEY_ZERO:
        LDRB R9, [R2, #1]
        STR R9, [R0]
        MOV R9, #0
        MOV R3, #0
        B MAIN

//Increase the display by 1
KEY_ONE:
        ADD R3, #1
        LDRB R9, [R2, R3]
        STR R9, [R0]
        MOV R9, #0
        B MAIN

//Decrease the display by 1
KEY_TWO:
        SUB R3, #1
        LDRB R9, [R2, R3]
        STR R9, [R0]
        MOV R9, #0
        B MAIN

//Clear the display and wait for a key press
KEY_THREE:
        LDRB R9, [R2]
        STR R9, [R0]
```

```
        MOV R3, #0

        //Wait for a key to be pressed
        WAIT_PRESS:
                LDR R7, [R1]
                CMP R7, #0
                BEQ WAIT_PRESS

        LDRB R9, [R2, #1]
        STR R9, [R0]

        //Wait for the reset key to be released
        WAIT_RELEASE:
                LDR R7, [R1]
                CMP R7, #0
                BNE WAIT_RELEASE

        MOV R9, #0
        B MAIN

END: B END

HEX_ADDR: .WORD 0xFF200020
KEY_ADDR: .WORD 0xFF200050
            .skip 1
HEX_BITS: .byte 0b00000000, 0b00111111, 0b00000110, 0b01011011, 0b01001111,
0b01100110, 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01100111, 0b0000000
                .skip 1

.end
```

Part 2

```
.global _start
_start:

//Store important things in memory to registers
MOV R0, #HEX_ADDR
LDR R0, [R0]
MOV R1, #EDGE_ADDR
LDR R1, [R1]
MOV R2, #HEX_BITS
MOV R3, #1 //Tens counter
```

```
MOV R4, #0 //Ones counter
MOV R6, #0 //Register to hold Edge Bit

//Set display to read 0
LDRB R9, [R2, #1]
STR R9, [R0]

MAIN:

        DO_DELAY:   LDR R7, =200000 // delay counter
        SUB_LOOP:   SUBS R7, R7, #1
                            BNE SUB_LOOP
        //Increment ones
        ADD R4, #1

        //Check to see if the edge bit is 1 (indicating if a key was pressed)
        BL CHECK_FOR_INTER

        //Increment Tens counter when R4 reaches 9
        CMP R4, #11
        BEQ ADD_TENS

        //Reset the counter when the counter reaches 99
        CMP R3, #11
        BEQ RESET

        //Display 10's and 1's
        LDRB R9, [R2, R3]
        LDRB R8, [R2, R4]
        LSL R9, #8
        ADD R9, R8
        STR R9, [R0]
        MOV R9, #0
        B MAIN

ADD_TENS:
        MOV R4, #0
        ADD R3, #1
        B MAIN

RESET:
        MOV R3, #1
        MOV R4, #1
```

```
        B MAIN

CHECK_FOR_INTER:
        PUSH {R6, R8, R9}

        //Check Edge Bit, if edge bit is not activated, exit subroutine via SKIP
        LDR R6, [R1]
        CMP R6, #1
        BGE INTER_TRUE
        B SKIP

        RETURN:
        //Wait for Edge Bit to be activated again to restart counter
        LDR R6, [R1]
        CMP R6, #1
        BLT RETURN
        STR R6, [R1]

        SKIP:
        POP {R6, R8, R9}
        BX LR

INTER_TRUE:
        //Clear Display, then reset Edge Bit
        LDRB R9, [R2]
        STR R9, [R0]
        STR R6, [R1]

B RETURN

END: B END

EDGE_ADDR: .WORD 0xFF20005C
HEX_ADDR: .WORD 0xFF200020
KEY_ADDR: .WORD 0xFF200050
            .skip 1
HEX_BITS: .byte 0b00000000, 0b00111111, 0b00000110, 0b01011011, 0b01001111,
0b01100110, 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01100111, 0b0000000
                .skip 1

.end
```

Part 3

```
.global _start
_start:

//Store important things in memory to registers
LDR R0, =HEX_ADDR
LDR R0, [R0]
LDR R1, =EDGE_ADDR
LDR R1, [R1]
LDR R2, =HEX_BITS
MOV R3, #1 //Tens counter
MOV R4, #0 //Ones counter
MOV R6, #0 //Register to hold Edge Bit

//Set display to read 0
LDRB R9, [R2, #1]
STR R9, [R0]

LDR R11, =0xFFFEC600
LDR R10, =20000000
STR R10, [R11]

LDR R11, =0xFFFEC608
MOV R10, #3
LDR R12, [R11]
ORR R10, R12
STR R10, [R11]

LDR R11, =0xFFFEC60C
STR R12, [R11]


MAIN:
        LDR R12, [R11]
        STR R12, [R11]

    DO_DELAY:
        LDR R12, [R11]
        CMP R12, #1
        BNE DO_DELAY

    //Increment ones
```

```
        ADD R4, #1

        //Check to see if the edge bit is 1 (indicating if a key was pressed)
        BL CHECK_FOR_INTER

        //Increment Tens counter when R4 reaches 9
        CMP R4, #11
        BEQ ADD_TENS

        //Reset the counter when the counter reaches 99
        CMP R3, #11
        BEQ RESET

        //Display 10's and 1's
        LDRB R9, [R2, R3]
        LDRB R8, [R2, R4]
        LSL R9, #8
        ADD R9, R8
        STR R9, [R0]
        MOV R9, #0
        B MAIN

ADD_TENS:
        MOV R4, #0
    ADD R3, #1
        B MAIN

RESET:
        MOV R3, #1
        MOV R4, #1
        B MAIN

CHECK_FOR_INTER:
        PUSH {R6, R8, R9}

        //Check Edge Bit, if edge bit is not activated, exit subroutine via SKIP
        LDR R6, [R1]
        CMP R6, #1
        BGE INTER_TRUE
        B SKIP

        RETURN:
        //Wait for Edge Bit to be activated again to restart counter
```

```
        LDR R6, [R1]
        CMP R6, #1
        BLT RETURN
        STR R6, [R1]

        SKIP:
        POP {R6, R8, R9}
        BX LR

INTER_TRUE:
        //Clear Display, then reset Edge Bit
        LDRB R9, [R2]
        STR R9, [R0]
        STR R6, [R1]

B RETURN

END: B END

EDGE_ADDR: .WORD 0xFF20005C
HEX_ADDR: .WORD 0xFF200020
TIMER_ADDR: .WORD 0xFFFEC600
KEY_ADDR: .WORD 0xFF200050
            .skip 1
HEX_BITS: .byte 0b00000000, 0b00111111, 0b00000110, 0b01011011, 0b01001111,
0b01100110, 0b01101101, 0b01111101, 0b00000111, 0b01111111, 0b01100111, 0b0000000
                    .skip 1

.end
```