```
;****************** main.s ***************
; Program initially written by: Yerraballi and Valvano
; Author: Vikram Pandian
; Date Created: 1/15/2018
; Last Modified: 1/18/2019
; Brief description of the program: Spring 2019 Lab1
; The objective of this system is to implement odd-bit counting system
; Hardware connections:
;  Output is positive logic, 1 turns on the LED, 0 turns off the LED
;  Inputs are negative logic, meaning switch not pressed is 1, pressed is 0
;    PE0 is an input
;    PE1 is an input
;    PE2 is an input
;    PE3 is the output
; Overall goal:
;   Make the output 1 if there is an odd number of 1's at the inputs,
;     otherwise make the output 0
; The specific operation of this system
;   Initialize Port E to make PE0,PE1,PE2 inputs and PE3 an output
;   Over and over, read the inputs, calculate the result and set the output

; NOTE: Do not use any conditional branches in your solution.
;       We want you to think of the solution in terms of logical and shift operations

GPIO_PORTE_DATA_R  EQU 0x400243FC
GPIO_PORTE_DIR_R   EQU 0x40024400
GPIO_PORTE_DEN_R   EQU 0x4002451C
SYSCTL_RCGCGPIO_R  EQU 0x400FE608

      THUMB
      AREA    DATA, ALIGN=2
;global variables go here
      ALIGN
      AREA    |.text|, CODE, READONLY, ALIGN=2
      EXPORT  Start
Start
    ; code to execute once at start goes here
    LDR R1, =SYSCTL_RCGCGPIO_R  ;activates clock for port E
    LDR R0, [R1]
    ORR R0, R0, #0x10               ; set bit 5 to turn on clock
    STR R0, [R1]
    NOP
    NOP
    LDR R1, =GPIO_PORTE_DIR_R       ;set DIR register
    LDR R0, [R1]                    ;need to set a 1 for output and 0 for input
    AND R0, R0, #0xF8
    ORR R0, R0, #0x8                ;this should make PE0-2 inputs and PE3 an output
    STR R0, [R1]

    LDR R1, =GPIO_PORTE_DEN_R       ;sets the enable register
    LDR R0, [R1]                    ;loads to register R0
    ORR R0, R0, #0x0F               ; sets to 1 which means enable
    STR R0, [R1]


loop
; code that runs over and over goes here

    LDR R2, =GPIO_PORTE_DATA_R      ;R2 now has data register address
    LDR R3, [R2]                    ;R3 now has the contents of the data register
    ;Use masks/shifts to find the individual bits of PE2-0
    ;Then XOR them to find the output PE3 bit value

    ;Initialize the masks
    AND R6, R6, #0
    ADD R6, R6, #1  ;mask for PE0

    AND R7, R7, #0
    ADD R7, R7, #0x02   ;mask for PE1

    AND R8, R8, #0
    ADD R8, R8, #0x04   ;mask for PE2
```

```
73
74      ;Now use the masks
75      AND R6, R3, R6  ;has PE0 val
76      AND R7, R3, R7  ;has PE1 val
77      AND R8, R3, R8  ;has PE2 val
78
79      ;need to shift the PE1 and PE2 bits to LSB
80      LSR R7, R7, #1
81      LSR R8, R8, #2
82
83      EOR R6, R6, R7
84      EOR R6, R6, R8 ;R6 should now have the value PE0 xor PE1 xor PE2 at its LSB
85
86      LSL R6, R6, #3  ;shifts bit to the 4th bit from the right (PE3)
87
88      STR R6, [R2]    ;stores to Data register
89
90
91        B    loop
92
93        ALIGN        ; make sure the end of this section is aligned
94        END          ; end of file
95
```