Homework #2
CSCE 4114/5114 Embedded Systems
Due Sept 7th, 2024

Problem #1 Short Answer: (60pts)

a) 10 pts. Write the C code to clear the bit specified by int bit position, in unsigned char data.

```
unsigned char data;

int bit_position;

int main(){
data &= ~(1<<bit_position);
}
```

b) 20 pts. Complete the C code to counts the number of bits set in as many iterations as there are set bits.

```
int main()
{
unsigned int n = 0; //Variable that set bits you want to count
unsigned int CountSetBits = 0; //Total number of bit set
printf("Enter the Number "); scanf("%d", &n); while(n)
{
    n = n & (n - 1);
    CountSetBits++;
}
```

c) 15 pts. C has a built in operator to shift but not rotate left or right. Shifting simply drops the end bit whereas a rotate takes the bit and places at the opposite end of the word. Complete the #define lines to implement rotating an integer left or right.

```
#include <stdio.h>
#define INT_BITS 32
#define ROTATE_LEFT(pos, data) (((data) << (pos)) | ((data) >>
(INT_BITS - (pos))))
 #define ROTATE_RIGHT(pos, data) (((data) >> (pos)) | ((data) <<
(INT_BITS - (pos))))
 int main() { int pos; // Number of rotation int data; //data which will be
rotate printf("%d Rotate Left by %d is ", data, pos); printf("%d \n",
ROTATE_LEFT(pos, data)); printf("%d Rotate Right by %d is ",data,
pos);
```

```
printf("%d \n", ROTATE_RIGHT(pos, data)); return
0;
}
```

c) 15 pts. Write the C code that swaps the values of a, b without using a temporary variable.

```
#include <stdio.h>
void SwapTwoNumber(int *a, int *b)
{
   *a = *a ^ *b;
   *b = *a ^ *b;
   *a = *a ^ *b;
}
```

Problem #2 GPIO Programming (40 pts)

a. List and describe the user accessible registers in the GPIO. –see data sheet

There are two user accessible registers in the GPIO. These consist of the GPIOx_DATA register and the GPIOx_TRI (3 state) register. The DATA register is used to read the input port and write to the output port. When a port is configured as input, writing to said port does not effect the DATA register. There are 2 DATA registers, however this depends if duel channel is configured on. If so then there are 2 DATA registers, else only 1 DATA register.

The TRI register configures ports as input or outputs depending on a bit. If the bit is set to 0 then the port is configured as output, else it is input. Similar to the GPIOx_DATA, this contains 2 TRI registers, however the second register is only present if dual channel is configured On.

b. Show the C to set up the GPIO and then read an integer from Port A and output the integer to Port B. Assume the base address of the GPIO is 0x40000000. You code snippet should include #define mask words you will write to configure Tristate registers, pointer addresses for the registers using offsets from the base address, code that sets the directions and a while(1) loop that performs the read from port A and write to port B.

```
#define GPIO_base 0x40000000 //address of base
#define outputDir 0x00000000 // All output bits
#define inputDir        0x0000001F // 5-input bits

int main()
{
  // Pointer definitions for GPIO
  // ** NOTE - integer definition causes offsets to be automatically be multiplied by 4!!
volatile int*base_GPIO      =  (int*)(0x40000000); /*GPIO Base */
volatile int *base_inGPIO  =(int*)(base_GPIO +0x0) /*Port A */
```

```
volatile int *tri_inGPIO                    = (int*)(base_GPIO +0x1)/*Port A Tristate*/   volatile
int *base_outGPIO =  int*)(base_GPIO +0x2)/*Port B */
volatile int *tri_outGPIO     = (int*)(base_GPIO +0x3)/*Port B Tristate*/

// setup Port A access

*tri_inGPIO = inputDir;


 // setup Port B access

*tri_outGPIO = outputDir;

//loop to read an input and sent to the output

While(1){
*base_outGPIO = *base_inGPIO;
}//end while
```