**Homework #2**

**CSCE 4114/5114 Embedded Systems**

**Due Sept 7th, 2024**

**Problem #1 Short Answer: (60pts)**

1. 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);

}

1. 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++;   
}

1. 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)**

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

1. 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 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_//address of base

#define outputDir \_\_\_\_\_\_\_\_\_\_\_\_\_\_// All output bits

#define inputDir \_\_\_\_\_\_\_\_\_\_\_\_\_\_// 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 = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\*GPIO Base \*/ volatile int \*base\_inGPIO = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\*Port A \*/ volatile int \*tri\_inGPIO = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\*Port A Tristate\*/ volatile int \*base\_outGPIO = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\*Port B \*/ volatile int \*tri\_outGPIO = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\*Port B Tristate\*/

// setup Port A access

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

// setup Port B access

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_;

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

While(1){

\*base\_outGPIO = \*base\_inGPIO;

}//end while