NYU TANDON SCHOOL OF ENGINEERING Name: ___________________________________

EL648 – Real Time Embedded Systems ID: ___________________________________

Quiz 1 Spring 2021

Question 1:

In class we showed the difference between passing by value and passing by reference using:

```
//By value
void swap( char x, char y)
{
…
}
//By reference
void swap (char * x, char *y)
{
…
}
```

a. Implement both versions of the swap functions using ARM assembly:

```
swapByVal       PROC
        ..
        ..
        ENDP
swapByRef       PROC
        ..
        ..
        ENDP
```

b. Write the C code required to call each of these functions using the variables sChar and dChar

Question 2:

Consider the following ARM assembly code segment. Describe what this function does and the contents of R0, R1, R2, and R3 for each iteration and when the function terminates. Also comment each line from __Main PROC to ENDP.

```
        AREA    myData, DATA
        ALIGN
str     DCB     “123”,0
        AREA    MyFunc, CODE
        EXPORT __main
        ALIGN
        ENTRY
 __main PROC
        LDR     r1, =str
        MOVS    r2, #0
loop    LDRB r0, [r1], #1
        CBZ     r0, stop
        CMP     r0, #0x30
        BLT stop
        CMP r0, #0x39
        BGT stop
        SUBS r0, r0, #0x30
        ADD r3, r2, r2, LSL #2
        ADD r2, r0, r3, LSL #1
        B loop
stop B  stop
        ENDP
```

Question 3:

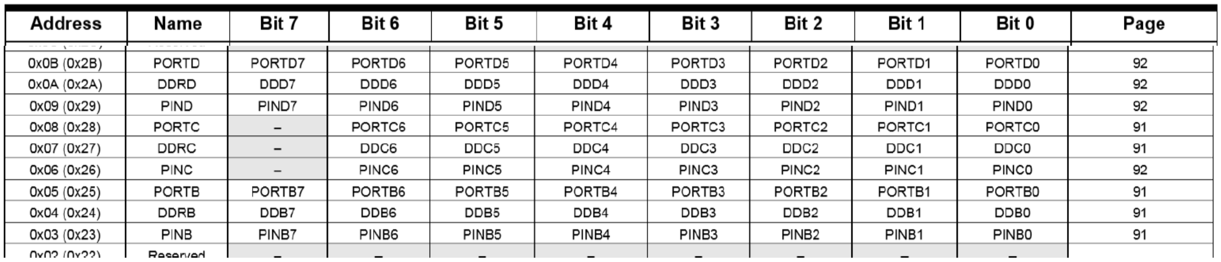
In class we showed how to simply light up an LED on our STM32 microcontroller. In fact, the procedure is similar for all processors including a simpler 8-bit ATmega 328P (a.k.a Arduino Uno).

For purposes of this exercise, let's say that the ATmega328 has 3 regular GPIO ports; B, C, and D, each with 8 pins. Every GPIO port has three registers (outlined in table below):

- **PORTX**: used to write output on a pin on port X that is configured as an output. When a pin is configured as an input, writing a 1 to its bit in PORTX activates a pull-up resistor for that pin, and writing a 0 to its bit in PORTX de-activates the pull-up resistor for that pin.
- **DDRX**: is the data direction register for port X. Writing a 0 to a bit in DDRX makes the corresponding port pin an input, while writing a 1 to a bit in DDRX makes the corresponding port pin an output.
- **PINX:** is the input register, used to read input data on port X. If a pin is configured as an output, then reading the input register bit corresponding to that pin will give you the value that was last output on the port.

Now see the following macro below:

```
#define PORTB (*(volatile unsigned char *)0x25)
```



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	92
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	92
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	92
0x08 (0x28)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	91
0x07 (0x27)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	91
0x06 (0x26)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	92
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	91
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	91
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	91
0x02 (0x22)	Reserved	–	–	–	–	–	–	–	–	

a. Using the chart above, write similar macros for PORTD, DDRD, and PIND. What is the purpose of the volatile keyword?
b. Write C code to configure PORTD pin 3 as output and write a 1 to it, without affecting the other pins on port D. (Two lines of code total)
c. Write C code to configure PORTD pin 6 as input (with a pullup resistor) without affecting the other pins on port D. (Two lines of code total)
d. Write a C main() function that sets up PORTD pin 1 as an input with pullup resistor and PORTD pin 2 as an output. The code should also blink an LED connected to pin 2 when pin 1 is high (one second period), otherwise the LED is off. (You can use wait_ms)
    a. If nothing is connected to pin 1, describe the behavior of the LED.
    b. What is the purpose of the pull up resistor?

Question 4:

Write the ARM Assembly code to flash all 4 user LEDs on your STM32 discovery board. Use a flashing frequency of 1 Hz). Recall, to set GPIO pins on our board, you need to configure the MODER, OSPEEDR, OTYPER, PUPDR and ODR registers for the port. Also, don't forget to connect the clock to GPIO! The following assembly code (or similar for other ports should be helpful).

```
RCC_AHB1ENR   EQU  0x40023830

GPIOD_MODER   EQU  0x40020C00

GPIOD_OTYPER  EQU  0x40020C04

GPIOD_OSPEEDR EQU  0x40020C08

GPIOD_PUPDR   EQU  0x40020C0C

GPIOD_ODR     EQU  0x40020C14

…
```