NYU TANDON SCHOOL OF ENGINEERING EL648 – Real Time Embedded Systems Quiz 1 Spring 2021	Name: Dhairya Upadhyay ID: DAU7593

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:
    mov r2,r0
    mov r0,r1
    mov r1,r2
    bx lr;

swapByRef:
   ldrb r2,[r0]
   ldrb r3,[r1]
   strb r3,[r0]
   strb r2,[r1]
   bx lr;
```

b. Write the C code required to call each of these functions using the variables sChar and dChar.
Ans.
- swapByVal (sChar , dChar);
- swapByRef (&sChar, &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
```

NYU TANDON SCHOOL OF ENGINEERING EL648 – Real Time Embedded Systems Quiz 1 Spring 2021	Name: Dhairya Upadhyay ID: DAU7593

```
        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
```

Ans.
The code starts at,
__main PROC
        LDR r1, =str //r1 is loaded with the memory location of the first byte in the string
        MOVS r2, #0 //r2 is equal to 0

Command/Iteration	First Iteration	Second Iteration	Third Iteration	Fourth Iteration
LDRB	r0 =*r1, r1=r1+1, ro = "1"	r0 =*r1, r1=r1+1, ro = "2"	r0 =*r1, r1=r1+1, ro = "3"	r0 =*r1, r1=r1+1, ro = 0x00
CBZ r0, stop	r0 = "1" so condition is not met	r0 = "2" so condition is not met	r0 = "3" so condition is not met	
CMP	Performs ro-#0x30 and sets C to 1	Performs ro-#0x30 and sets C to 1	Performs ro-#0x30 and sets C to 1	
BLT	Not executed	Not executed	Not executed	
CMP	r0=r0-#0x39, N flag sets to 1	r0= "2"-#0x39, N flag sets to 1	r0= "3"-#0x39, N flag sets to 1	
BGT	Not executed	Not executed	Not executed	
SUBS	r0 = r0 - #0x30 = #0x01	r0 = "2" - #0x30 = #0x02	r0 = "3" - #0x30 = #0x03	
ADD	r3 = r2 + r2<<2 = 0	r3 = r2 + r2<<2 = 5	r3 = r2 + r2<<2 = 60	
ADD	r2 = r0 + r3 <<1 = r0 =1	r2 = r0 + r3 <<1 = r0 =12	r2 = r0 + r3 <<1 = r0 =123	

NYU TANDON SCHOOL OF ENGINEERING EL648 – Real Time Embedded Systems Quiz 1 Spring 2021	Name: Dhairya Upadhyay ID: DAU7593

B	Go back to loop	Go back to loop	Go back to loop	Since the condition is met, it goes to stop

So, the final values in the register are as follows:
R0 = 0x00
R1 = one byte after str
R2 = 123
R3 = 60

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?
Ans.
define PORTD (*(volatile unsigned char *)0x2B)
define DDRD (*(volatile unsigned char *)0x2A)
define PIND (*(volatile unsigned char *)0x29)

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)
Ans.
DDRD |= (1 << 3)

```
PORTD |= (1 << 3)
```

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)
Ans.

```
DDRD &= ~(1 << 6)
PORTD |= (1 << 6)
```

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)
Ans.

```
void setup(){
        DDRD &= ~(1 << 1);
        PORTD |= (1 <<1);
        DDRD |= (1 <<2);
}
void loop(){
        while(TRUE){
                if ((PIND >> 1) & 1){
                        PORTD |= (1 << 2);
                        sleep(1000);
                }
        }
        PORTD &= ~(1 << 2);
}
```

a. If nothing is connected to pin 1, describe the behavior of the LED.
Ans.
The LED keeps blinking.

b. What is the purpose of the pull up resistor?
Ans.
The pull up resistor is used tto ensure a known state for a signal.

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
…
```

NYU TANDON SCHOOL OF ENGINEERING EL648 – Real Time Embedded Systems Quiz 1 Spring 2021	Name: Dhairya Upadhyay ID: DAU7593

Ans.
Helper.s

```
.syntax unified

.global asm_blink
.global asm_function

.equ DELAY_INTERVAL, 0x30C008
.equ RCC_AHB1ENR, 0x40023830
.equ GPIOD_MODER,   0x40020C00
.equ GPIOD_OTYPER,  0x40020C04
.equ GPIOD_OSPEEDR, 0x40020C08
.equ GPIOD_PUPDR,   0x40020C0C
.equ GPIOD_ODR, 0x40020C14

asm_function:

    LDR     R1, =RCC_AHB1ENR
    LDR     R0, [R1]
    ORR.W   R0, #0x08
    STR     R0, [R1]

    LDR     R1, =GPIOD_MODER
    LDR     R0, [R1]
    ORR.W   R0, #0x55000000
    AND.W   R0, #0x55FFFFFF
    STR     R0, [R1]

    LDR     R1, =GPIOD_OTYPER
    LDR     R0, [R1]
    AND.W   R0, #0xFFFF0FFF
    STR     R0, [R1]

    LDR     R1, =GPIOD_OSPEEDR
    LDR     R0, [R1]
    AND.W   R0, #0x00FFFFFF
    STR     R0, [R1]

    LDR     R1, =GPIOD_PUPDR
    LDR     R0, [R1]
    AND.W   R0, #0x00FFFFFF
    STR     R0, [R1]

    BX      LR;

asm_blink:

    turnON:
        LDR     R1, =GPIOD_ODR
```

NYU TANDON SCHOOL OF ENGINEERING EL648 – Real Time Embedded Systems Quiz 1 Spring 2021	Name: Dhairya Upadhyay ID: DAU7593

```
        LDR     R0, [R1]
        ORR.W   R0, #0XF000
        STR     R0, [R1]

        LDR     R2, =DELAY_INTERVAL

    delay1:
        CBZ     R2, turnOFF
        SUBS    R2, R2, #1
        B       delay1

    turnOFF:
        LDR     R1, =GPIOD_ODR
        LDR     R0, [R1]
        AND.W   R0, #0xFFFF0FFF
        STR     R0, [R1]
        LDR     R2,=DELAY_INTERVAL

    delay2:
        CBZ     R2, delayDone
        SUBS    R2, R2, #1
        B       delay2

    delayDone:
        B turnON;
```

Main.c

```
#include <mbed.h>
#include <USBSerial.h>

void test();
extern "C" void asm_blink();
extern "C" void asm_function();

int main() {

  // put your setup code here, to run once:

  while(1) {
    // put your main code here, to run repeatedly:
    test();
    wait_ms(100);
  }
}

void test(){
  asm_function();
  asm_blink();
}
```