NYU TANDON SCHOOL OF ENGINEERING Name: ______________________________

EL648 – Real Time Embedded Systems ID: ______________________________

Exam 1 - Fall 2022

Instructions: Answer all questions on this paper. You may use any reference materials from class or from our class websites and/or reference sheets. Any cheating identified will result in a zero score on this exam.

Question 1:

In class we showed how to use "mixed" languages to code our embedded system. Suppose you would like to write an assembly function called "GetReg" which takes in a single parameter as an index to a general purpose register. For example, 0 corresponds to R0, 6 corresponds to R6 etc.)

The function then returns the value of that register as a 32 bit integer. In essence, GetReg() is a helper function to get the values of any general purpose register.

a. Write the arm assembly code required to implement the function GetReg() and expose it to your C code. Be sure to handle out-of-range values.
b. Write the C code to import the assembly function GetReg()and call the assembly function to return the values of R0 through R15 and print them to the serial terminal.
c. Submit both your .cpp and.S file.

Question 2:

Repeat problem 1, but now modify your code segments so that the function GetReg() takes in one parameter, rArray, which is the address of an array of 16 integers. The GetReg() function should populate the rArray with the contents of all 16 general purpose registers, in order. For example, rAarray[0]=R0, rArray[1]=R1 etc.) In addition, the function should return the contents of the CPSR register.

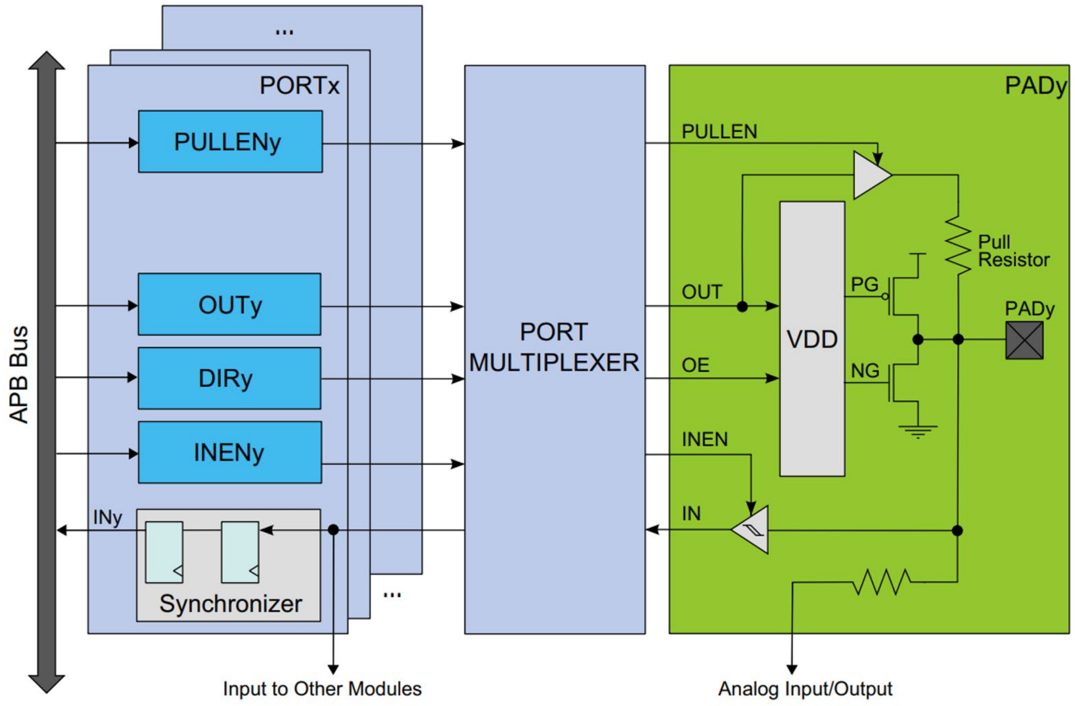
Question 3:

Consider the following ARM assembly code segment (*in italics*). Describe what this function does and the contents of R0, R1, R2, and R3 for each iteration and when the function terminates.

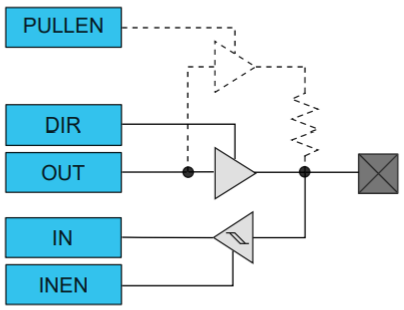
```
        AREA    myData, DATA
        ALIGN
str     DCB     "123",0
        AREA    MyFunc, CODE
        EXPORT MyFunc
;;;;;;;;;;;;;;;;;;;;;;;;;
 MyFunc:
        LDR     r1, =str
        MOVS    r2, #0
loop:
        LDRB r0, [r1], #1
        CBZ     r0, stop
        CMP     r0, #0x30
        BLT stop
        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 4:

In class we demonstrated that by controlling a few registers, we can configure our GPIO for several different applications. The example in our notes shows the configurations for the AtmelSAMD21 (M0+) 32 bit ARM microcontroller (see slide 12). A snap shot is shown below:



Using the datasheet for the ATSAMD21 microcontroller (easily found online and in this assignment), write a c function configIO() that configures the PULLEN,OUT,DIR, and INEN so that PortA pins 0-16 are set to Inputs with pull-up resistors, and pins 17-31 are setup as totem-pole output (input enabled) as seen below. See Section 23 of the datasheet. You can use any HAL macros or definitions of registers if you wish.


PULLEN
DIR
OUT
IN
INEN