

Final Exam**Date:** May 16, 2022

UT EID: _____

Printed Name: _____
Last, First

Your signature is your promise that you have not cheated and will not cheat on this exam, nor will you help others to cheat on this exam:

Signature: _____

Instructions:

- Closed book and closed notes. No books, no papers, no data sheets (other than the last two pages of this Exam)
- No devices other than pencil, pen, eraser (no calculators, no electronic devices), please turn cell phones off.
- Please be sure that your answers to all questions (and all supporting work that is required) are contained in the space (boxes) provided. Do Not write answers on back of pages
- You have 180 minutes, so allocate your time accordingly.
- Unless otherwise stated, make all I/O accesses friendly.
- Please read the entire exam before starting.

Problem 1	15	
Problem 2	10	
Problem 3	10	
Problem 4	15	
Problem 5	10	
Problem 6	15	
Problem 7	10	
Problem 8	15	
Total	100	

[15 points] Problem 1: Fundamentals. Answer the following short questions in the boxes provided.

1. (5 pts) Consider this piece of C code

```
const uint32_t *pt;  
uint32_t Operate(const uint32_t y){  
    static uint32_t z=0;  
  
    z++;  
  
    return y+z;  
  
}
```

- A) Permanent in RAM
- B) Permanent in nonvolatile ROM
- C) Temporary in R0
- D) On the stack
- E) None of the above

Answer the following with one letter A – E (see above for meaning) assuming AAPCS.

- a. (2pts) Where is the variable **pt** located?

- b. (1 pt) Where is the parameter **y** located?

- c. (2 pts) Where is the variable **z** located?

2. (2 pts) You performed **UART1_ICR_R=0x00000010;** in Lab 9. What does this do?

3. (3 pts) Assume memory contains byte values given below (left). Assume R0 = 0x20000002. What will be the value in R1 if the instruction given on the right is executed?
(Give your answer in hexadecimal.)

0x20000000 contains 0x44
0x20000001 contains 0x53
0x20000002 contains 0x62
0x20000003 contains 0x87
0x20000004 contains 0x98
0x20000005 contains 0xA2

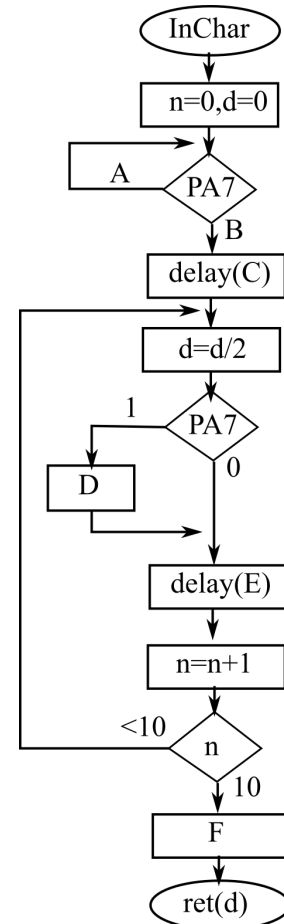
LDRSH R1,[R0]

R1:

4. (5 pts) It is possible to communicate between two TM4Cs using the UART protocol without the UART hardware. It is done by following a strict protocol of writing bits to a GPIO pin with proper timing between bit writes if you are implementing the sender or bit reads if you are implementing the receiver. This is called bit-banging. The flowchart below is one possible algorithm to receive 8 bits of data, with some missing pieces that you have to choose from the options given. The C prototype for this function is `uint8_t InChar(void);`

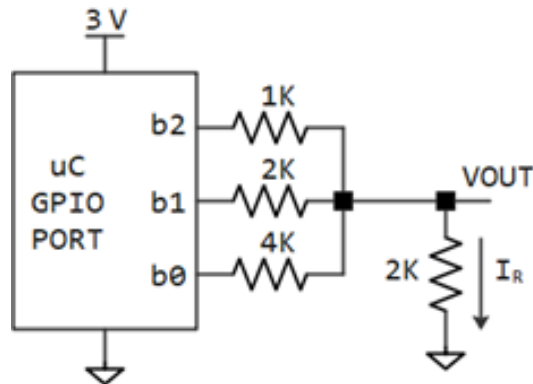
The **InChar()** function uses the GPIO input pin, PA7. The protocol is 1 start bit, 8 data bits, and 1 stop bit (just like UART). You want to read the PA7 in the middle of the bit-time. The bit time is 10 ms. There is a function called **delay(t)** with input parameter **t** that waits exactly **t** ms. This **InChar()** function, when called, will wait for an incoming frame, accept the 10 bits, and return the 8-bit data value for the frame. Choose the best answers 1 2 3 or 4 for each box.

- AB 1) A=0; B=1
2) A=1; B=0
3) A none; B=0 or 1
4) other, specify the correct answer
- C 1) 0 ms
2) 5 ms
3) 10 ms
4) other, specify the correct answer
- D 1) d = d+1
2) d = d+0x80
3) d = d+0x200
4) other, specify the correct answer
- E 1) 0 ms
2) 5 ms
3) 10 ms
4) other, specify the correct answer
- F 1) d = d&0x01FE
2) d = d&0x00FF
3) d = (d/2)&0x00FF
4) other, specify the correct answer



[10 points] Problem 2: DAC and circuits.

An engineering student who has taken EE 319K built the following DAC circuit. The analog **VOUT** is a linear function of the digital output of the microcontroller, but this DAC works differently than the DAC in Lab 6. Assume that a logic-low results in 0V, and a logic-high results in 3.0V for this microcontroller.



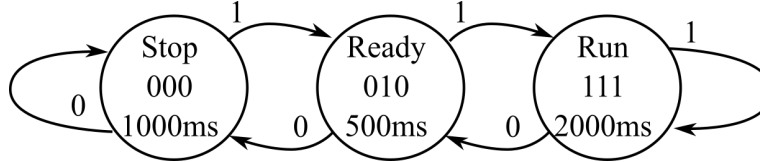
Answer the following

- a. (3 pts) What is **VOUT** when the digital output is 0x1? Show your work.

- b. (3 pts) What is **VOUT** when the digital output is 0x7? Show your work.

- c. (4 pts) For a particular digital output, the measured current through the **2KΩ** load resistor, **I_R**, is between 0.6mA and 0.8mA. What was the digital output that resulted in this measurement? Show your work.

[10 points] Problem 3: Finite State Machine. Consider the following FSM state transition graph (STG). The inputs and outputs in the STG are given in binary.



1. (7 pts) Complete the missing parts in the C implementation below. You do not implement the **doOutput** and **getInput** functions; these two functions are given.

<pre> enum StateNum {Stop, Ready, Run}; typedef enum StateNum SNum_t; struct State { // 1-bit input uint32_t Out; // 3-bit output uint32_t Time; // 1ms units [] Next[[]];}; typedef const struct State State_t; State_t FSM[3]={ { [], [], { [] }}, { [], [], { [] }}, { [], [], { [] }} }; </pre>	<pre> int main(){ [] CS = Stop; while(1){ uint8_t in; doOutput(FSM[CS].Out); delay(FSM[CS].Time); in = getInput(); CS = [] ; } } </pre>
---	---

2. (3 pts) Assuming the TM4C is running at 16 MHz, complete the following **delay** routine that was called to delay for **t** milliseconds. Assume SysTick is already initialized to run at 16 MHz without interrupts.

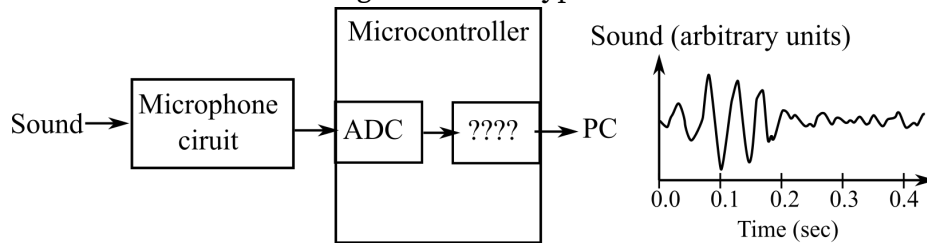
<pre> void delay(uint32_t t){ for (uint32_t i=0; i < t; i++){ NVIC_ST_RELOAD_R = [] ; NVIC_ST_CURRENT_R = 0; while ((NVIC_ST_CTL_R & []) == 0){} } } </pre>
--

[10 points] Problem 4: ADC, Interrupts, and Sampling

A microphone circuit has been hooked up to a stethoscope to form an electronic stethoscope that is used to measure fetal heart sounds.

- Fetal heart sounds have a frequency range of 50 to 200 Hz
- The analog microphone circuit behaves linearly and has a full-scale range of 20 Pascals (Pa). When the sound level is 0 Pa, the microphone circuit outputs 0V. When the sound level is 20 Pa, the circuit outputs 3.3V. When the sound level is 10 Pa, the circuit outputs 1.65V.

The output of the circuit is connected to the ADC on the microcontroller. One audible sound occurs with each heartbeat. The figure shows a typical sound:



The sound level will be stored in the computer as an 8-bit decimal fixed point number with resolution 0.1 Pa. The integer portion of the sound level should be stored in the following global

```
uint8_t Isound; // 0.1 Pa
```

- (2 pts) What is the minimum sampling rate that should be used to properly capture all signal content from a fetal heart?

- (2 pts) According to Valvano's postulate, what sampling rate should be used if you want to accurately recreate a graphical plot of the heart sounds being measured?

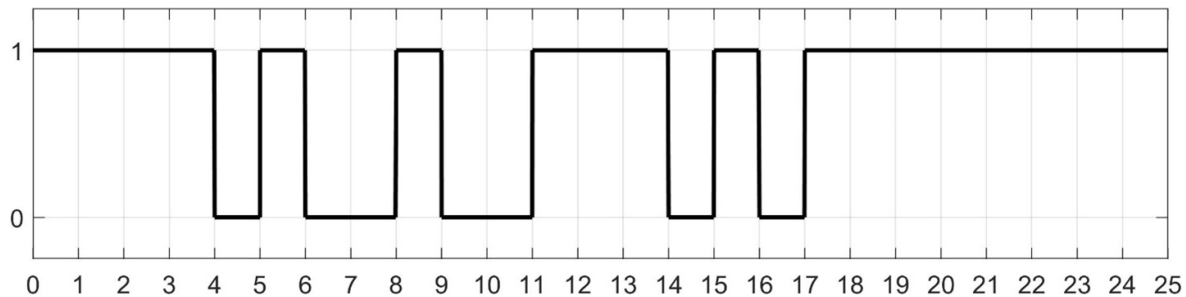
- (3 pts) What device would you use to communicate data to a PC, labeled as ??? in figure

- (10 pts) Write the SysTick ISR in C that will perform the operations identified in comments below. The ADC has been initialized in 12-bit mode using sequencer 3, with software start. SysTick has been initialized to interrupt at the proper rate. The communication device you defined in question #3 has also been initialized. **Do not call any functions**, show all code needed. Comments to the right give you *what* needs to be done, you write the *code* to do it!!

	// Start a conversion of the ADC
	// Poll the ADC conversion status // and wait until ADC is complete // using busy-wait
	// Read the 12-bit ADC value from // the sequencer 3 FIFO and // acknowledge the ADC
	// Convert the ADC value to iSound
	// Send the iSound to the PC // using busy-wait

[10 points] Problem 5: Communications/UART

An engineering student who has not taken EE 319K needs your help in identifying the data sent by a serial device. The student connected a logic analyzer to the UART output pin of the device and observed the following signal, which contains exactly **three frames**. Each frame has 1 start, **n** data bits, and 1 stop. There are no idle periods between the three frames. The time units on the graph are in ms.



Help the student decode the data message by answering the following questions.

1. **(4 pts)** What is the bit-time?

--

2. (1 pts) What is **n** (5 bits, 6 bits, 7 bits, or 8 bits)?

--

3. **(5 pts)** Decode the message as three hex values in the order they were sent by the device:

Frame 1 =	
Frame 2 =	
Frame 3 =	

[15 points] Problem 6: Stack You are given the following C code:

<pre>typedef struct nums { uint32_t a; uint32_t b; uint32_t c; uint32_t sum; } nums_t; void sum_nums(nums_t *inp){ (*inp).sum = (*inp).a + (*inp).b + (*inp).c; }</pre>	<pre>static uint32_t result; int main() { nums_t test; test.a = 0x50600000; test.b = 0x03040000; test.c = 0x00001020; test.sum = 0; sum_nums(&test); result = test.sum; }</pre>
--	--

Answer the following questions:

1. (5 pts) The main function has a local variable called `test`, which is of type `nums_t` and is allocated on the stack. Assuming the initial value of the SP at the beginning of main is `0x20000320`, show the contents of the stack. The struct is written to stack with its attribute (a) at the top of the stack. Leave blank any values that are unknown. **Mark the SP after test is placed on stack.** Following AAPCS guidelines.

Address	Value (displayed in hex as uint32)
0x20000310	
0x20000314	
0x20000318	
0x2000031C	
0x20000320 (initial SP)	

2. (4 pts) The function `sum_nums` is called by main passing the address of the local variable `test` to the function. In which register does the function receive this address, and what are the contents of this register?

Register:	Contents:
-----------	-----------

3. (6 pts) Convert the `sum_nums` function to assembly. It is partially done for you:

<pre>sum_nums push { LDR R4, [, #0] ; R4 = (*inp).a pop {</pre>

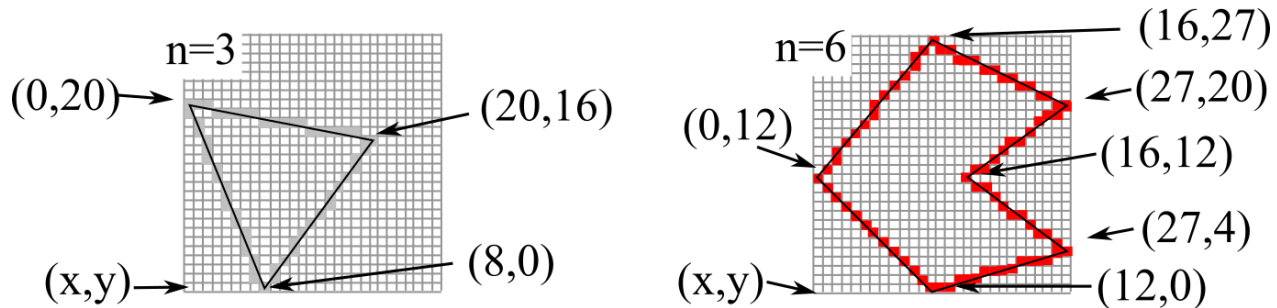
[10 points] Problem 7: Stack in C

You will complete the implementation of a stack data structure where each element of the stack is a 16-bit unsigned integer. Hint: This works similar to assembly *push* and *pop* instructions with the top serving as the Stack Pointer.

<pre>// Variables and Constants #define N 200 // Size of Stk #define Fail 0 #define Success 1 uint16_t Stk[N]; // Stack Storage // pointer to topmost element uint16_t *top; // Initialize the Stk to empty // Input: None // Make Stk empty void Stk_Init(){ // Initialize top <div style="border: 1px solid black; height: 80px; margin-top: 10px; padding: 5px;"> top = _____ ; </div> } </pre>	<pre>// Push an item on the Stack // Input: data has item to pushed // Output: Success or Fail int Stk_Push(uint16_t data){ <div style="border: 1px solid black; height: 60px; margin-top: 10px; padding: 5px;"> // check full if (_____) { return Fail; } // Push data onto stack (2 lines) <div style="border: 1px solid black; height: 60px; margin-top: 5px;"> <div style="border-bottom: 1px solid black; height: 30px;"></div> <div style="height: 30px;"></div> </div> return Success; } // Pop an item from the Stack // Input: dataptr pointer to empty place // into which data is to be popped // Output: Success or Fail int Stk_Pop(uint16_t *dataptr){ <div style="border: 1px solid black; height: 60px; margin-top: 10px; padding: 5px;"> // check Empty if (_____) { return Fail; } // Pop from stack, write to *dataptr // with top value from stack (2 lines) <div style="border: 1px solid black; height: 60px; margin-top: 5px;"> <div style="border-bottom: 1px solid black; height: 30px;"></div> <div style="height: 30px;"></div> </div> return Success; } } </div></div></pre>
---	---

[15 points] Problem 8: Programming, Data structures, and Design.

The goal is to implement a sprite system that can represent closed polygons. Each polygon has an (x,y) position specifying the location of the lower left corner of the sprite within the LCD coordinate space. The vertices of the polygon are defined as a variable length array of points, relative to the (x,y) position. Polygons have 3 to 10 vertices, life, and color. The figure below shows examples of two polygons of 3 and 6 vertices.



The sprite data structure is fixed and should not be changed.

```
struct point{
    int32_t dx; // location relative to sprite position x
    int32_t dy; // location relative to sprite position y
};
typedef struct point point_t;

struct sprite{
    int32_t x,y;      // lower left corner of sprite
    uint32_t n;       // number of vertices in polygon
    uint16_t col;     // color for all lines
    point_t vert[10]; // 3 to 10 vertices, relative to (x,y)
    int life;         // 0 for dead, 1 for alive
};
typedef struct sprite sprite_t;
```

```
sprite_t Poly[100];
```

Write the draw function that draws all sprites that are alive. You are given a function that draws a line from (x1,y1) to (x2,y2) of color c:

```
Line(int32_t x1, int32_t y1, int32_t x2, int32_t y2, uint16_t c);
```

Hint: A polygon with **n** vertices requires you to draw **n** lines total. You will need **all** attributes of a `sprite_t` struct to draw the polygon it represents.

```
// Draws all sprites that are alive in the PoLy (global) array  
void Draw(void){
```

```
}
```