ECE319K Fall 2023 Exam1 Version A

UT EID: ______________________ First: _______________ Last: ______________________

Instructions:

- Closed book and closed notes. No books, no papers, no data sheets (other than the addendum)
- 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. *Anything outside the boxes/blanks will be ignored in grading.* You may use the back of the sheets for scratch work.
- You have 75 minutes, so allocate your time accordingly.
- For all questions, unless otherwise stated, find the most efficient (time, resources) solution.
- Unless otherwise stated, make all I/O accesses friendly and all subroutines AAPCS compliant.
- *Please read the entire exam before starting.*

(15) Question 1.

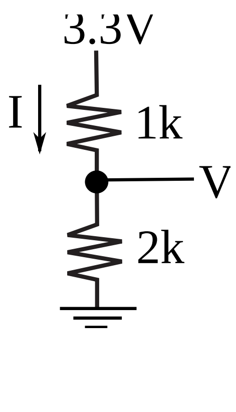
(3) Part a) What does **volatile** mean in context of computer memory ………

(3) Part b) If R1 is 0xFFFFFFFF. Does the branch to **Happy** occur? ………

```
CMP R1,#3
BHS Happy
```

(3) Part c) Considering R0 as input and R1 as output, what is the mathematical relationship between R1 and R0? ………………………

```
LSLS R1,R0,#2
ADDS R1,R1,R0
```



(3) Part d) What is V? …………

(3) Part e) What is I? ……………

(15) Question 2a. There are two 100-element 8-bit signed global arrays, **X Y**.

<pre>   .data
X: .space 100
Y: .space 100
      .text
main: LDR R0,=X
      LDR R1,=-10
      BL   Fill
// fills X with -10
      LDR R0,=Y
      MOVS R1,#10
      BL   Fill
// fills Y with +10
loop: B   loop</pre>
	<pre>int8_t X[100];
	int8_t Y[100];
	void Fill(int8_t buf[], int8_t data){
	  for(int i=0; i<100; i++){
	    buf[i] = data;
	  }
	}
	int main(void){
	  Fill(X,-10);
	  Fill(Y,10);
	  while(1){};
	}</pre>

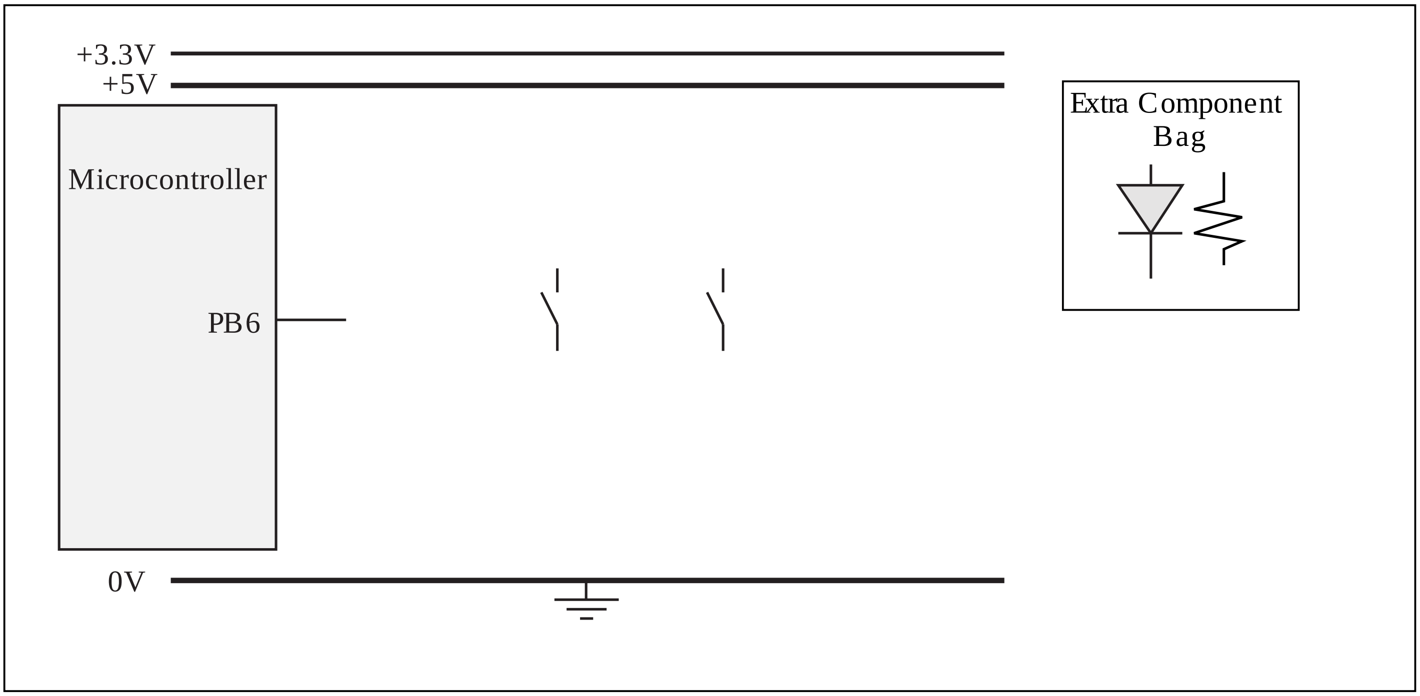
Write a Cortex M assembly subroutine implementation of C function **Fill** above. Follow AAPCS.

```
Fill:
```

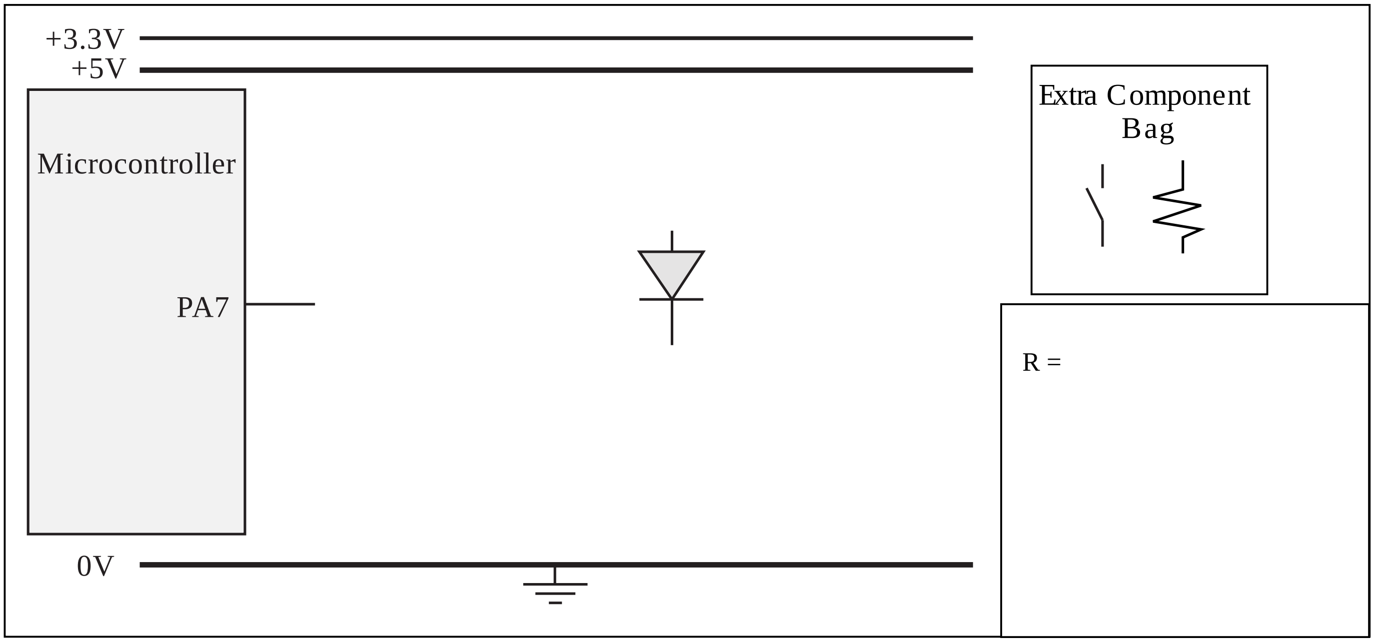
(20) Question 3a. Write a C function to implement factorial. Return 0xFFFFFFFF (4,294,967,295) if the calculation would overflow 32-bit unsigned math. Note that 12! is about 479 million, while 13! is about 6 billion. The function prototype is fixed and cannot be changed. **n**! is defined as 1*2*3*…***n**. 0! is 1.

```
uint32_t Fact(uint32_t n){
```

(10) Question 4a. Interface two **switches** to Port B. The voltage on PB6 should be 3.3V if both switches are pressed, and PB6 should be 0V if zero or one switch is pressed. You may use one or more elements from the bag. Specify resistor values if resistors needed. Connect these components to +5V, +3.3V, 0V, and the microcontroller as needed.



(10) Question 5a. Interface an **LED** to Port A pin 7 using negative logic. The desired LED operating point is 1.5V, 3mA. The microcontroller output high voltage is 3.1V, the microcontroller output low voltage is 0.3V. For any resistor(s) you use, show your work for determining the resistor value(s). You may use one or more elements from the bag.



(10) Question 6a. Show the contents of all five registers after we execute this code.

```
MOVS R0,#0
MOVS R1,#1
MOVS R2,#2
MOVS R3,#3
MOVS R4,#4
PUSH {R2}
PUSH {R1,R3,R4}
LSLS R0,R3,R1
POP  {R4}
POP  {R1,R2,R3}
```

R0 =

R1 =

R2 =

R3 =

R4 =

(5) Question 7a. Assume there is an array pointed to by R0. MSPM0 architecture is little endian. *Hint: look carefully at the memory addresses in the following figure.*

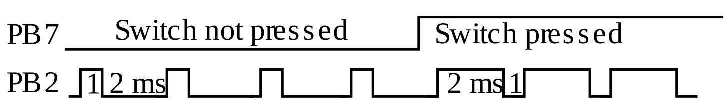
Address	Contents	
0x20201000	**0x90**	**<- R0**
0x20201001	**0x91**	
0x20201002	**0x92**	
0x20201003	**0x93**	
0x20201004	**0x94**	
0x20201005	**0x95**	

Assume register R0 equals 0x20201000. What is the value of R2 after executing the following instructions?

```
MOVS  R1,#2
LDRSH R2,[R0,R1]
```

R2=

(15) Question 8a. PB7 is a positive logic switch input, and PB2 is a positive logic LED output. The initialization function is given, which sets PB7 to input and PB2 to output. **There are 3 bugs in the following solution. Circle the 3 bugs and make corrections to fix the bugs.** You are given a delay function, **Delayms**, which waits 1ms. You must follow AAPCS.



The main loop should repeat these steps over and over.

Read the switch

If the switch is pressed and held, make the LED a 66% duty cycle 333 Hz wave

Turn on, wait 2 ms, turn off, wait 1 ms.

If the switch is not pressed, make the LED a 33% duty cycle 33 Hz wave

Turn on, wait 1 ms, turn off, wait 2 ms.

```
main: MOVS R0,#0
      BL   Clock_Init80MHz // 12.5ns
      BL   Init // given, do not write
      LDR  R5,=GPIOB_DIN31_0
      LDR  R6,=GPIOB_DOUTSET31_0
      LDR  R7,=GPIOB_DOUTCLR31_0
      MOVS R4,#0x02
      MOVS R2,#0x80

loop: LDR  R1,[R5]
      BICS R1,R1,R2
      BEQ  low

high: STR  R4,[R6]
      BL   Delayms
      BL   Delayms
      STR  R4,[R7]
      BL   Delayms
      B    loop

low:  STR  R4,[R6]
      BL   Delayms
      STR  R4,[R7]
      BL   Delayms
      BL   Delayms
      B    loop
```