ECE319K Fall 2024 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.*

(20) Question 1a.

(6) Part a) Specify the op code for **XXX** to perform the equivalent C operation. **z** is signed and in R3,
The assembly instructions are:

```
MOVS R4,#0x03
XXX  R3,R3,R4
```

XXX	C Operation
ASRS	z = z / 8;
MULS	z = z * 3;
EORS	z = z ^ 0x03;
ORRS	z = z | 0x03;
BICS	z = z & (~0x03);
ANDS	z = z & 0x03;

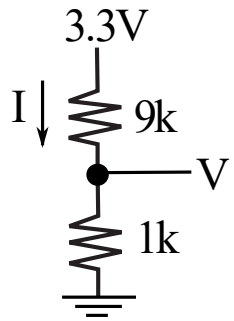
(3) Part b) The initial value of **z** is 17=0x11=0b10001
What is the resulting value of **z** after all 6 lines of C are executed? …

17/8=2, 2*3=6, 6^3=5, 5|3=7, 7&(~3)=4, 4&3=0

(5) Part c) What is the value of R1 after these instructions are executed in hexadecimal? ……

```
        .text
        .align 2
Data:   .byte 0x50,0x60,0x70,0x80,0x90,0xA0,0xB0,0xC0
main:    LDR  R2,=Data
         LDRH R1,[R2,#4]
```

Two bytes are 0x90 and 0xA0, in little endian 0xA090, or 0x0000A090



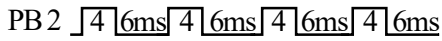
(3) Part d) What is V? …….

3V*1k/10k = 0.33V

(3) Part e) What is I? ……

3.3V/(1+9k) =0.33mA

(5) Question 2a. There is a positive logic LED connected to PB2. The LED voltage is 2V and the current is 5mA. The software creates this output.



How much power, in mW, is delivered to the LED? Show your work.

Full power is 2V*5mA = 10 mW
Duty cycle is H/(H+L) = 40%
LED power is 0.4*10mW = 4mW

(5) Question 3a) Choose one op code to implement the corresponding C. Assume **x** is unsigned.

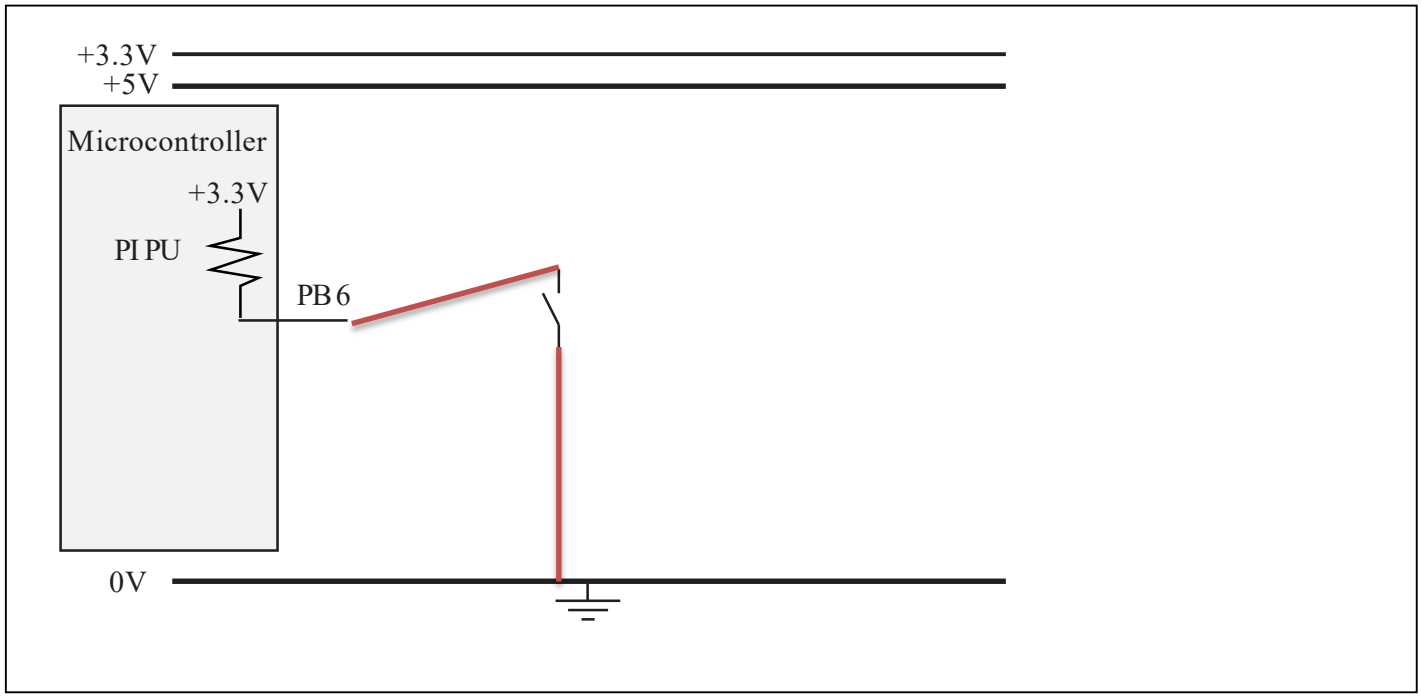
```
    LDR R0,=x
    LDR R1,[R0]
    CMP R1,#10
    XXX Skip
    BL  Thing
Skip:
```

XXX	C Operation
BNE	`if(x==10){Thing();}`
BLS	`if(x>10){Thing();}`
BHI	`if(x<=10){Thing();}`

(15) Question 4a) Convert the assembly to C box by box. All values are 32-bit unsigned. The assembly code follows AAPCS. Give the local variable in R1 the name **m**. Ignore overflow.

Assembly	C
`Fun:` `MOVS R1,#1 // R1 is m`	`uint32_t Fun(uint32_t n){` `uint32_t m=1;`
`Loop: MULS R1,R1,R0`	`do{` `m = m*n;`
`SUBS R0,R0,#1`	`n--;` `}`
`CMP R0,#0` `BNE Loop`	`while(n != 0);`
`MOVS R0,R1` `BX LR`	`return m;` `}`

(10) Question 5a. Interface a **switch** to PB6 using NO resistors. To interface the switch without resistors, the software activates internal pull up, **PIPU**. I.e., there is an internal resistor from PB6 to 3.3V. Connect the switch to +5V, +3.3V, 0V, and the microcontroller as needed.



(14) Question 6a. Assume a negative logic LED is interfaced to PB16. Write assembly function that turns on the LED. Make the code friendly.

```
LED_On:
 LSLS R0,R0,#16
 LDR  R1,=GPIOB_DOUT31_0 // address
 LDR  R2,[R1]            // previous value
 LDR  R3,=(1<<16)        // bit 16 mask
 EORS R0,R0,R3           // negative logic LED requires 0 output
 BICS R2,R2,R3
 ORRS R0,R0,R2
 STR  R0,[R1]            // do output
 BX   LR
```

```
LED_On:
        LDR   R3,=(1<<16)        // bit 16 mask
        CMP   R0,#0
        BNE   clr     // R0=0 turns on LED
        LDR   R1,=GPIOB_DOUTSET31_0 // turn off LED
        B     done
clr:    LDR   R1,=GPIOB_DOUTCLR31_0 // turn on LED
done:   STR   R3,[R1]            // do output
        BX    LR
```

(6) Question 7a Consider debugging the flashing LED in Lab 2. For each debugging procedure, classify as N M or H

N = nonintrusive means LED flashes at the same rate with or without debugging,
M = minimally intrusive, means LED flashes just a little bit slower with debugging
H = highly intrusive, LED flashes at a rate MUCH slower with debugging

Part a) Logic analyzer connected to all GPIO pins ----------	N
Part b) Breakpoints ----------	H
Part c) Dump GPIO data into a buffer ----------	M
Part d) Scope connected to GPIO output pin ----------	N
Part e) Black box testing: running code and looking at the LED with your eyes --	N
Part f) Single stepping ----------	H

(10) Question 8a. Show the contents of the stack at the marked points respectively in the execution of the following code. The initial stack pointer is **0x20201008**.

```
0x00002000      MOVS R0,#0
0x00002002      MOVS R1,#1
0x00002004      PUSH {R1,R0}
0x00002006      MOVS R4,#4
0x00002008      BL   Func
0x0000200A      POP  {R0,R1}
…
            Func:
0x00002042      PUSH {R4,LR}
0x00002044      MOVS R4,#10
0x00002046      MULS R0,R0,R4 // <----A) draw stack
0x00002048      POP  {R4,PC}
```

Give value of SP and the contents of the stack after execution point A):

0x20200FF8	4
0x20200FFC	0x200B
0x20201000	0
0x20201004	1
0x20201008	???
0x2020100C	???
0x20201010	???
0x20201014	???
0x20201018	???

SP= 0x202000FF8

Place ??? in any box for which you do not know its contents, or is not guaranteed to be valid

(15) Question 9a. Write a C function that searches an array and returns the minimum value. If the array is empty return 255. Each element is 8-bit unsigned. The arrays are terminated with sentinel 255. The prototype cannot be changed

```
uint8_t Min(uint8_t buf[]);
uint8_t buffer1[5]={4,3,7,8,255};
uint8_t buffer2[8]={7,6,5,4,3,2,1,255};
uint8_t buffer3[1]={255};
int main(void){
  uint8_t ans1 = Min(buffer1); // answer is 3
  uint8_t ans2 = Min(buffer2); // answer is 1
  uint8_t ans3 = Min(buffer3); // answer is 255
  while(1){}
}

uint8_t Min(uint8_t buf[]){
  uint8_t ans = 255; int i=0;
  while(buf[i] != 255){
    if(buf[i] < ans){
     ans = buf[i];
    }
    i++;
  }
  return ans;
}
```