Name: ____

(10) Question 1)

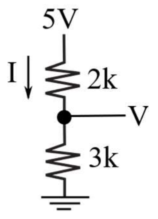
(2) Part a) Assume the following memory contents, R0=0x00000102, and R1=0x20200000. What is in R2 after this one instruction executed? Show your answer in hexadecimal.

```
LDRH R2,[R0,R1]
```

0x20200100	0x80
0x20200101	0x81
0x20200102	0x82
0x20200103	0x83
0x20200104	0x84

(2) Part b) Assume R1 has a signed 32-bit value, write assembly code that divides R1 by 16, placing the result back in R1.

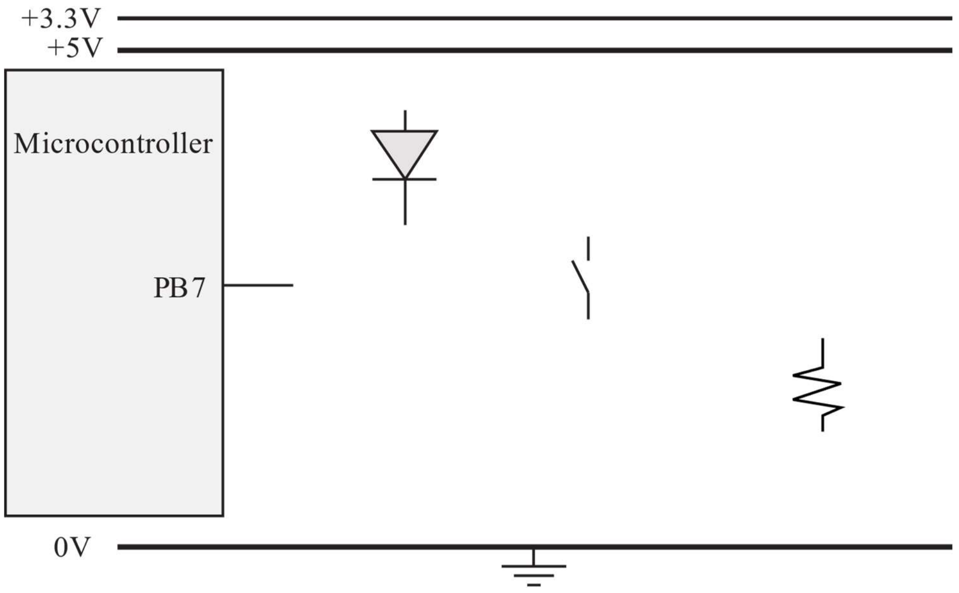
(2) Part c) Write assembly code to swap the values of R2 and R5 using just PUSH and POP.



(2) Part d) What is V?

(2) Part e) What is I?

(12) Question 2. Interface a **switch** to PB7 using **negative logic**. Show all connections needed and values for any resistors needed. You are not using internal pull-up or pull-down.



(6) Question 3.

(2) Part a) Assume **x, y, z** are integer variables, **x** is 3 and **y** is 2. What is **z** after this line is executed?

```
z = (x<<y)^10;
```

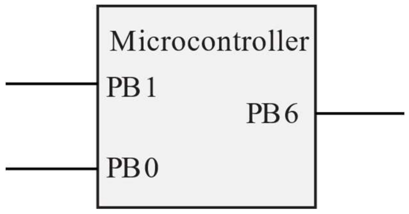
(2) Part b) Assume you have a delay function that can wait any integer number of bus cycles. The period of the PWM wave must be 10,000 bus cycles. How many different duty cycles can you make?

```
while(1){
  LED_On();
  Wait(H); // time high
  LED_Off();
  Wait(L); // time low
}
```

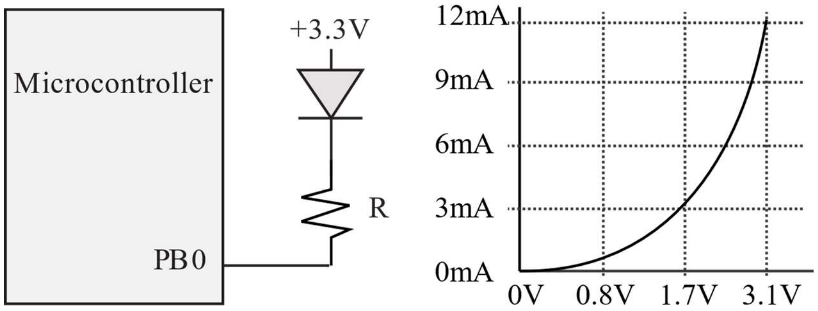
(2) Part c) What is the range of the **uint16_t** data type in C? Give both the smallest and largest possible values. You can leave it as an expression like $2^{10} + 1$.

Smallest =	Largest =

(15) Question 4. Write **assembly code** to do the following in a loop (i.e., run over and over without stopping). Read from two GPIO input pins (PB0 and PB1), and then write to one GPIO output pin (PB6). If the two input values are the same, output a ONE, else output a ZERO. The output to PB6 must be friendly. You do not know the configuration or status of the other bits on Port B. You can assume that Port B has been reset and powered on, the PINCM registers have been appropriately initialized, and that PB6 has been output enabled. You may freely use R0-R7 without pushing/pulling them on the stack.



(12) Question 5. Consider the LED circuit below interfaced with PB0.



(2) Part a) Is this LED interface positive or negative logic?

(3) Part b) Suppose the pin output is such that the LED is on. We want to choose a value for R. In no more than 5 words each, explain what would happen for the values of R below.

1. R = 100k ohms
2. R = 500 ohms
3. R = 1 ohm

(5) Part c) Suppose the desired operating point of the LED is 1.7V, 3mA. The output high voltage (V_{OH}) of the microcontroller is 3.1V, and the output low voltage (V_{OL}) of the microcontroller is 0.1V. Derive an equation and solve for the correct R?

(2) Part d) If you choose a resistor with a value twice as large as the one calculated in c) will the LED be brighter or dimmer?

(15) Question 6) Consider the following code. What does the code do, in seven words or less? *Hint*: try hand-executing with small values for **a** and **b**.

```
uint32_t mystery(uint8_t a, uint8_t b){
  uint32_t result = 0;
  uint32_t msb = 1 << 7;
  for (int32_t i = 0; i < 8; i=i+1){
    result = result << 1;
    if ((a & msb) != 0){
      result = result + b;
    }
    a = a << 1;
  }
  return(result);
}
```

Below is a direct translation of the mystery function, with assembly instructions and operands missing (indicated by boxes). Fill in the missing assembly instructions and operands. Your completed assembly code should be a direct translation of the C version (not just get the same result). You are only allowed to fill in the blanks; you cannot add any additional instructions or change any of the instructions or arguments. It follows AAPCS.

```
Mystery:  PUSH {R4-R7,LR}
L1:  MOVS R4, #0
L2:  LDR  R2, =1<<7
L3:  MOVS R3, ______
L4:  CMP  R3, ______
L5:  ______  L14
L6:  ADDS R4, ______ , ______
L7:  MOVS R5, R2
L8:  ANDS ______ , ______ , R0
L9:  BEQ  ______
L10: ADDS R4, R4, R1
L11: ______  R0, #1
L12: ADDS R3, #1
L13: B    ______
L14: ______ R0, R4
L15: POP  {R4-R7,PC}
```

(8) Question 7) You are given a C function that outputs to Port A

```
void Output(uint32_t data){
  GPIOA->DOUT31_0 = data;
}
```

You write a subroutine called **MyAssemblyFunction** that calls **Output** with the **data** parameter equal to 5. Follow AAPCS.

```
 .global Output
MyAssemblyFunction:
```

(7) Question 8: Assume the following register values:

R0	0
R1	1
R2	2
R3	3
R13(SP)	0x20201000
R14(LR)	0x000001FF

Draw the stack after these two instructions are executed. Each box has a 32-bit value.

```
   PUSH {R3}
   PUSH {LR,R1}
```

	0x20200FF4
	0x20200FF8
	0x20200FFC
	0x20201000
	0x20201004
	0x20201008
	0x2020100C

What is the SP after these two instructions are executed?

(15) Question 9. A variable-length character string is allocated 10 spaces and defined as follows

```
        .data
String: .space 10
```

You may not add any additional global variables. The string should remain null terminated. I.e., there should always be a 0x00 at the end of the string. Therefore, there is space for 9 characters. You may assume all bytes of the string are initialized to 0x00 once at the start of the system, Therefore, the string is initially empty. Write **an assembly language** function called **`Append`**, which appends one 8-bit character to the end of the string each time **`Append`** is called. The 8-bit value to store is passed in the lower 8-bits of Register R0. If the string already contains 9 characters, any call to **`Append`** will not store. Furthermore, if the data is 0x00, do not append. I.e., only nonzero characters will be saved. Follow AAPCS. The following shows what happens if your function is called twice

```
  Append('h'); // String is "h"  or 0x68,0x00
  Append('i'); // String is "hi" or 0x68,0x69,0x00
```

```
Append:
```