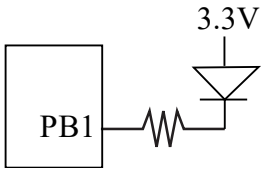
UT EID: _____ First: _____ Last: _____

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 120 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.*

(5) Question 1) SysTick interrupts are armed and running. PB1 is initialized as an output.

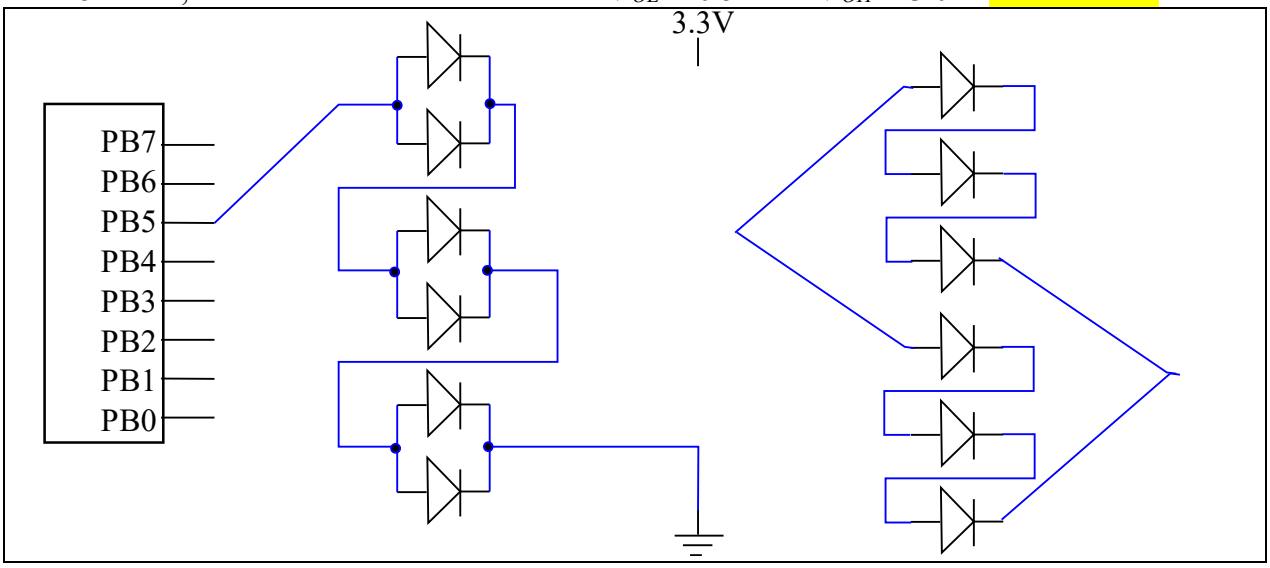
```
void SysTick_Handler(void){
  GPIOB->DOUT31_0 = GPIOB->DOUT31_0^0x02;
  if(GPIOB->DOUT31_0 & 0x02){
    SysTick->LOAD = 10000;
  }else{
    SysTick->LOAD = 30000;
  }
  SysTick->VAL = 0; // causes SysTick to reload
}
```



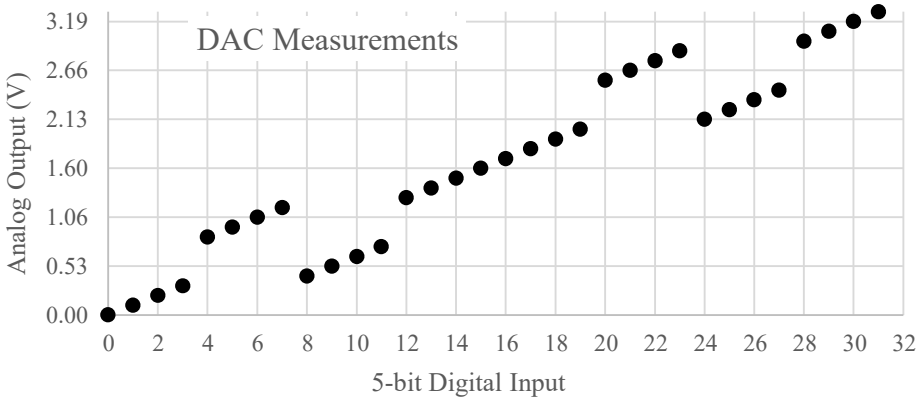
The resistor is the correct value to activate the LED. What will be the approximate brightness of the LED? Give your answer as a percentage from 0% (off) to 100% (full).

On for 30000, off for 10000, duty cycle is about 75% brightness

(5) Question 2) You are given six identical LEDs. Each LED parameter is $I_d = 2.5$mA, $V_d = 1.0$V. Interface all LEDs to PB5 **without any resistors**. If PB5 is high, then all six LEDs should be on. If PB5 is low, then all six LEDs should be off. V_{OL} is 0.5V and V_{OH} is 3.0V. Two answers



(5) Question 3) A student built a 5-bit binary weighted DAC. The circuit shows how the resistors were supposed to be connected. The student collected this calibration data. Two resistors have been swapped. Which two resistors were swapped, creating this result?



Enter your answer A-E into the box.
These voltages were created Vout=3.3V*(16*B4+4*B3+8*B2+2*B1+B0)/31
Think about how to swap the 4,5,6,7 outputs with the 8,9,10,11 outputs
Swap 001xx ⇔ 010xx, C) swap 3k with 6k

(5) Problem 4. A device is connected to ADC1 channel 0.
The way to think about the ADC is compare it to lab.
Lab: distance goes from 0 to 2cm
Final: Acceleration goes from -4 to +4 g

Lab: transducer maps 0 to 2cm into 0 to 3.3V
Final: transducer maps -4 to +4 g into 0 to 3.3V

Lab: ADC goes from 0 to +3.3V into 0 to 4095
Final: ADC goes from 0 to 2.5V into 0 to 4095

Lab: transducer+ADC maps 0 to 2cm into 0 to 4095
Final: transducer+ADC maps -4 to +4 g into 0 to 4095

Lab: we averaged multiple samples to improve SNR
Final: we add (not divide) two samples to improve SNR

Lab: transducer+ADC+average maps 0 to 2cm into 0 to 4095
Final: transducer+ADC+sum maps -4 to +4 g into 0 to 8190 (13-bit sum)

Lab: decimal fixed point represented 0 to 2cm as 0 to 2000 (delta=0.001cm)
Final: binary fixed point represented -4 to +4 g into -4096 to 4096 (delta=(1/1024)g)

Lab : software converted 0 to 0; and 4095 to 2000

```
Distance = 2000*ADC_In()>>12;
```

Final: software converted 0 to -4096; and 8190 (close to 8192) to +4096

```
int32_t accel; // integer portion of fixed-point number
void TIMG12_IRQHandler(void){
  static int32_t prev = 0; // previous ADC, 0 to 4095
  //solution
  if((TIMG12->CPU_INT.IIDX) == 1){ // this will acknowledge
    int32_t data = ADC1_In(); // 0 to 4095
    int32_t sum = prev+data;  // 0 to 8190
    accel = (data + prev) - 4096;
    //store the adc value for next cycle
    prev = data;
  }
//solution end




}
```

(5) Problem 5. If the device from Question 4) is accelerated at 2g, answer the five questions.

Part a) What is the voltage output of the device in volts?

*6/8 * 2.5V = ¾ * 2.5V = 15/8= 1.875V*

Part b) What is the ADC value from calling **ADC1_In**?

1.875V/2.5V * 4096 = *(15/8)/(5/2)= 30/40 = (3/4) *4096=3072*

Part c) What is the fixed-point gravity value in the variable **accel**?

2048

Part d) According to the **Nyquist Theorem** what are the minimum and maximum frequencies that can be reliably captured given the system defined in Question 4?

0≤f<50 Hz

Part e) According to the **Valvano Postulate** what is the maximum frequency that can be reconstructed and still look like the original signal?

10 Hz

(10) Question 6) This system is written entirely in assembly and DOES NOT comply with AAPCS. Show the assembly implementation of **Divideby32** that modifies the 16-bit signed global in place by effectively dividing the global by 32. Do not round the result. E.g., 63/32 is 1.

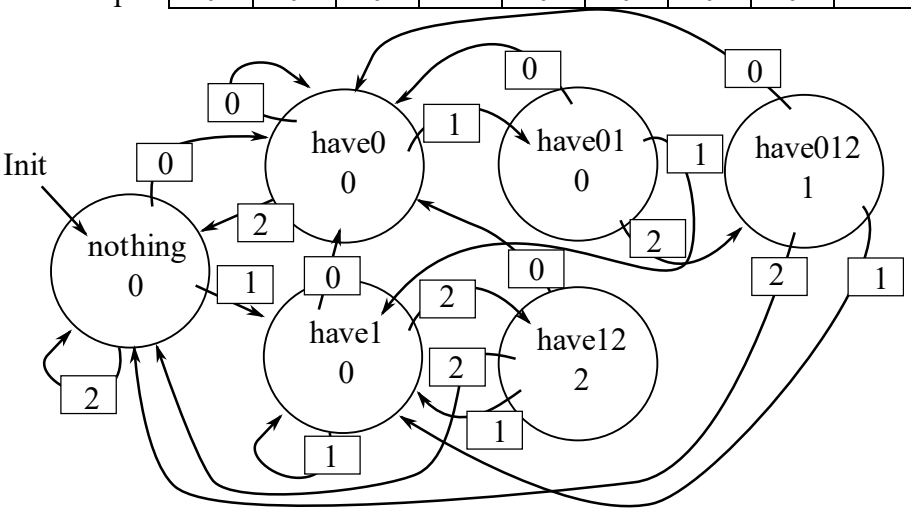
```
    .data                          .text                          .text
    .align 2                       LDR  R7,=I                      LDR  R6,=J
I:  .space 2 // signed             PUSH {R0,R7}                    PUSH {R0,R6}
J:  .space 2 // signed             MOVS R7,#0                      MOVS R6,#0
                                   BL   Divideby32                 BL   Divideby32
                                 // I=I/32                       // J=J/32
                                   ADD  SP,#8                      ADD  SP,#8
```

The one main program invokes **Divideby32** for **I**, and a second main invokes it for **J**. Notice the main programs balance the stack; **Divideby32** must also balance the stack. **Divideby32** has no input or output parameters in registers; rather, **Divideby32** must use the parameter passed on the stack. *Hint*: draw a stack picture and look at what is being passed.

```
Divideby32:
  LDR   R0,[SP,#4] // address of I or J
  MOVS  R2,#0
  LDRSH R1,[R0,R2] // contents of global
  ASRS  R1,#5      // global/32
  STRH  R1,[R0]    // global = global/32I
  BX    LR

  LDRSH R1,[R0]    // is not correct
// You must use ASRS and not implement divide with a loop
  POP {R0,R7} // R7=address of I or J ***** weird but works
  MOVS  R2,#0
  LDRSH R1,[R7,R2] // contents of global
  ASRS  R1,#5      // global/32
  STRH  R1,[R7]    // global = global/32I
  PUSH {R0,R7} // balance stack
  BX    LR
```

(5) Question 7) Draw the FSM STG for the following system. The input can be 0 to 2. There are no time delays. The initial output is 0. If the input sequence is 0,1,2 the output is 1. However, if the input sequence is 1,2 the output is 2. For example,

Input	2	3	1	2	3	3	0	1	2	3	1	2	3
Output	0	0	0	2	0	0	0	0	1	0	0	2	0



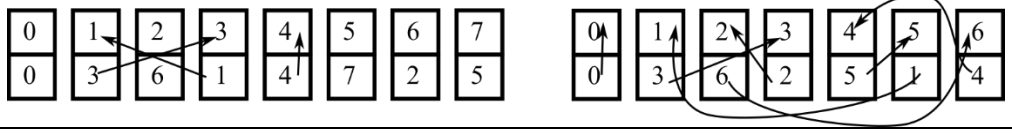
(15) Problem 8. Write a function to create a ring of nodes:
(3) Part a) Define a struct **node_t** with two fields: An unsigned 8-bit integer **x**, and an unsigned 8-bit integer **next**, which contains the index of another node

```
#define N 8
    struct node {
        uint8_t x;
        uint8_t next;
    };
    typedef struct node node_t;
```

(2) Part b) Create an array **Nodes** of **N node_t** structs. **N** can change if recompiled.

```
node_t Nodes[N];
```

(5) Part c) Write code to initialize the **Nodes** array, such that the value **x** of each node equals its index value, and each **next** is equal to 3***x** modulo **N**. One is N=8, the other is N=7.




```
void InitializeRings(void){

  for(uint8_t idx=0; idx<N; idx++) {
     Nodes[idx].x = idx;
     Nodes[idx].next = (3*idx)%N;
  }
}
```

(5) Part d) Write code to trace a ring of nodes starting with the node whose **x** value is **start**, and then moving to the node whose **x** value is given by next. Do this repeatedly until you return to **start**. Your function should return the length of this ring. For example, if **N** = 8, and **start** = 1, length = 2 (1->3->1), If **N**=7, **start** = 4, length = 7

```
uint8_t TraceRing(uint8_t start){

    uint8_t length = 1;
    uint8_t nextVal = Nodes[start].next;
    while(nextVal != start) {
      nextVal = Nodes[nextVal].next;
      length++;
    }
    return length;
}
```

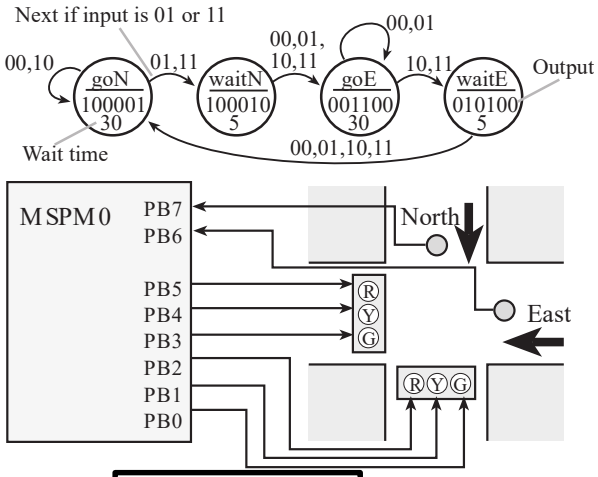
(15) Question 9. If you are in 319K solve the C problem, if you are in 319H solve the C++ problem. Below is code to implement a FIFO. This code supports using all the FIFO data space, meaning it does not waste one location to differentiate between full and empty. Fill in the blanks in the code. You are not allowed to change any variable definitions. C and C++ are same

```
static int *data;
static int head,tail,x,size;

// pt is pointer to an array
// s is size of array
int Init(int *pt, int s){
  size = s;
  data = pt;
  tail = head = 0;

  [ x ] = [ 0 ] ;

  return(1); // success
}

int Put(int value){
  if( [ x ] < [ size ] ){
    data[tail] = value;
    tail = (tail+1)%size;
    [ x++ ] ;
    return(1); // success
  } else {
    return(0); // fail
  }
}
int Get(void){
  if( [ x ] == [ 0 ] ){
    return(-1);
  } else {
    int ret = data[head];
    head = (head+1)%size;
    [ x-- ] ;
    return(ret);
}}
```

```
class fifo_t {
 private:
  int *data;
  int head,tail,x,size;
 public:
  bool Init(int size);
  bool Put(int value);
  int  Get();
};
bool fifo_t::Init(int s){
  size = s;
  data=(int *)malloc(4*size);
  if(data==NULL)return(false);
  tail = head = 0;
  [ x ] = [ 0 ] ;
  return(true); // success
}
bool fifo_t::Put(int value){
  if( [ x ] < [ size ] ){
    data[tail] = value;
    tail = (tail+1)%size;
    [ X++ ] ;
    return(true); // success
  } else {
    return(false); // fail
  }
}
int fifo_t::Get(){
  if( [ x ] == [ 0 ] ){
    return(-1);
  } else {
    int ret = data[head];
    head = (head+1)%size;
    [ x-- ] ;
    return(ret);
}}
```

(15) Question 10) You are given a **Delay** function written in C, which follows AAPCS. It has this prototype **`void Delay(uint32_t sec)`**. The parameter **sec** is the number of seconds to wait. Complete the **main** program that runs this FSM. You need not write **Delay** or **Traffic_Init**. The initial state is **goN**. Your output must be friendly.

```
 .global Delay
goN:    .long 0x21
        .long 30  //30 sec
        .long goN,waitN,goN,waitN
waitN:  .long 0x22
        .long 5   //5 sec
        .long goE,goE,goE,goE
goE:    .long 0x0C
        .long 30  //30 sec
        .long goE,goE,waitE,waitE
waitE:  .long 0x14
        .long 5   //5 sec
        .long goN,goN,goN,goN
```



Part a) Which register should you use for **Rx**? R0

Part b) Which register should you use for **Ry**? R7

```
main: BL   Traffic_Init
      LDR  Ry,=goN
      LDR  R4,=GPIOB_DOUT31_0
      LDR  R5,=GPIOB_DIN31_0
loop:
      LDR  Rx,[Ry,#0 ]
      LDR  R1,[R4]
      MOVS R6,#0x3F
      BICS R1,R1,R6
      ORRS Rx,Rx,R1   // ADDS ok
      STR  Rx,[R4]
      LDR  Rx,[Ry,#4 ] // shift right by 6 to get index,
// shift left by 2 to get offset
      BL   Delay
      LDR  Rx,[R5]
      MOVS R1,#0xC0
      ANDS Rx,Rx,R1
      LSRS Rx,Rx,#4
      ADDS Rx,Rx,#8
      LDR  Ry,[R7,R0]    same as [Ry,Rx] [Rx,Ry] [R0,R7]
      B    loop
```