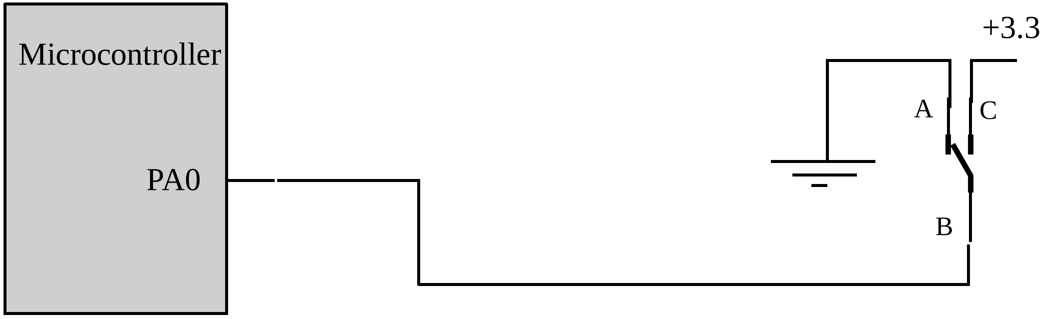
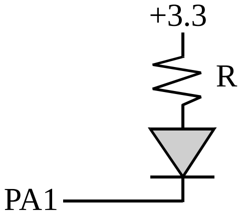
(10) Question 1. Consider a game that has 100 bouncing balls.

```
void SameSpace(void){ unsigned long i,j;
  for(i=0; i<99; i++){
    for(j=i+1;j<100;j++){
      if((Ball[i].x == Ball[j].x)&&(Ball[i].y == Ball[j].y)){
        Ball[i].angle = (Ball[i].angle+90)%360;
        Ball[j].angle = (Ball[j].angle+90)%360;
      }
    }
  }
}
```

(5) Question 2. Interface a single-pole double-throw (SPDT) switch to input port PA0



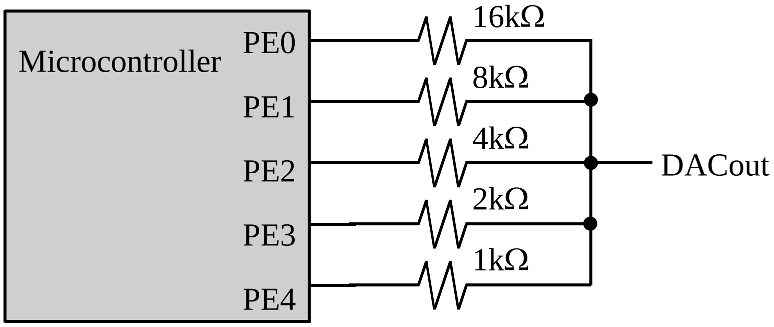
(5) Question 3. Interface an LED to PA1. R= (3.3-1.2-0.1)/2mA = 2V/2mA = 1000Ω.



(8) Problem 4. Implement a C function outputs a string to UART0.

```
void UART_OutString(unsigned char *pt){
  while(*pt){
    while((UART0_FR_R & 0x20) != 0){};    // Wait until TXFF = 0
    UART0_DR_R = *pt;      // output
    pt++;                  // next
  }
}
```

(8) Question 5. Design a 5-bit DAC using the binary-weighted configuration. Any set of resistor values that doubles is ok. Choose values in 1k to 1M range.



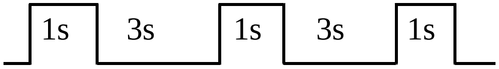
(6) Question 6. Add C code to define the following variables
v1 should be a public permanently-allocated 32-bit signed variable
v2 should be a temporary 32-bit unsigned variable private to the function **Fun_Init**
v3 should be a permanently-allocated 16-bit signed variable private to the function **Fun_Init**
v4 should be a permanently-allocated 16-bit signed variable, private to the file **Fun.c**.

```
// This is the first line of the Fun.c code file
long v1;          // public permanent
static short v4;  // private to file, permanent

void Fun_Init(int in){     // code
unsigned long v2; // private to function, temporary
static short v3;  // private to function, permanent


}
// this is the last line of the Fun.c code file
```

(10) Question 7. Show an **assembly subroutine** that sets each element of the buffer to its index value. Assuming **i** varies from 0 to 99, set **Buffer[i] = i;**

```
Fill LDR R0,=Buffer    ;pointer to buffer
     MOV R1,#0         ;index
     MOV R2,#100       ;ending index
loop STR R1,[R0]       ;put index into buffer
     ADD R1,#1         ;next index
     ADD R0,#4         ;next address
     CMP R1,R2
     BLO loop
     BX  LR
```

(10) Question 8. Write C or assembly code that creates this output on PA2 using SysTick interrupts.



Part a) Show the initialization code that runs once

```
volatile unsigned long Counts = 0;
#define PA2          (*((volatile unsigned long *)0x40004010))
void SysTick_Init(void){
  SYSCTL_RCGC2_R |= 0x01;    // activate port A
  Counts = 0;
  GPIO_PORTA_DIR_R |= 0x04;   // make PA0 out
  GPIO_PORTA_DEN_R |= 0x04;   // enable digital I/O on PA0
  NVIC_ST_CTRL_R = 0;         // disable SysTick during setup
  NVIC_ST_RELOAD_R = 4999999; // reload value
  NVIC_ST_CURRENT_R = 0;      // any write to current clears it
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x40000000;
  NVIC_ST_CTRL_R = 0x07;      // enable,source,arm
  EnableInterrupts();
  PA2 = 0x04;                 // PA2 initially high
}
```

Part b) Show the SysTick ISR

```
// Executed every 100ms
void SysTick_Handler(void){
  Counts = Counts + 1;
  if(Count == 10){
    PA2 = 0x00;             // PA2 now is low
  }
  if(Count == 40){
    PA2 = 0x04;             // PA2 now is high
    Count = 0;
  }
}
```

(10) Question 9. State the term that is best described by each definition.

Part a) An address that specifies the location of an interrupt service routine. **vector**

Part b) A type of computer architecture where data is read from memory in the same way machine codes are fetched from memory. **von Neumann**

Part c) The theorem that says the frequency at which the ADC is sampled must be higher than the frequency of the signal being sampled. **Nyquist**

Part d) An interfacing approach where the hardware causes a specific software routine to be executed. **interrupts**

Part e) A debugging technique that stores strategic information into an array at run time, and the contents of the array are observed afterwards. **dump**

Part f) A term that describes a variable specifying whether some or all of the software has access to the variable. Hint: the answer is not private, and the answer is not public. **scope**

Part g) A measure of software size, specifying how many bytes of memory are required for the software. **Static efficiency**

Part h) A software step that explicitly clears the trigger flag. ---------------------- **acknowledge**

Part i) The name given to describe 1,048,576 bytes. --------------------------------- **mebibyte**

Part j) A type of digital logic where the output is either zero or off. ---------------- **Open collector**

(4) Question 10. The Stellaris LM3S1968 has a 0 to 3V 10-bit ADC. What will be the digital output of the ADC if the input voltage is 0.75 V? **1024*0.75/3 = 256**

(2) Question 11. If R0 equals -10, what will be in register R0 after executing these instructions?

```
   LSL  R1,R0,#3  ; R1 is -80 (times 8)
   ADD  R0,R0,R1  ; R0 is -80 + -10 = -90
```

This is a multiply by 9 operation, works with signed or unsigned numbers

(6) Question 12. Consider a SysTick ISR.

Part a) 8 registers are pushed R0,R1,R2,R3,R12,LR,PC,PSW

Part b) Since LR = 0xFFFFFF9, it pops the 8 registers R0,R1,R2,R3,R12,LR,PC,PSW

(10) Question 13. A distance is represented as unsigned binary fixed-point number with resolution of 2^{-4} cm. Calculate the *cost* = (1.5 dollars/cm)**distance*. The cost is represented as an unsigned decimal fixed-point number with resolution of $0.01. The function should return the variable integer representing cost in Register R0. For example if the distance is 1.25 cm. The cost will be (1.5 dollars/cm)*1.25 cm = $1.87 (or $1.88 depending on how you round).

Part a) Let *I* be the variable integer representing *distance*. Give an equation relating *distance* and *I*?

$\textit{distance} = I * 2^{-4}$ cm

Part b) Let *J* be the variable integer representing *cost*. Give an equation relating *cost* and *J*?

$\textit{cost} = J * \$0.01$

Part c) Write the assembly subroutine that converts distance to cost. Start with the desired operation

$\textit{cost} = (1.5 \text{ dollars/cm}) * \textit{distance}$

$J * \$0.01 = (1.5 \text{ dollars/cm}) * I * 2^{-4}$ cm

$J = 150 * I / 16$

Multiply first and divide second

```
CalculateCost
     MOV  R1,#150
     MUL  R0,R0,R1   ;150*I
     LSR  R0,R0,#4   ;150*I/16
     BX   LR
```

(6) Question 14. (a) is 8 because R4 and R5 are on top. (b) is 1 because this is an 8-bit FIFO, (c) is 4 because we are deallocating 1 word, 4 bytes.

```
pt       EQU  8    ;??(a)??
Fifo_Get PUSH {R0} ;allocate local
         PUSH {R4,R5}
         LDR  R0,=PutPt
         LDR  R0,[R0]
         LDR  R1,=GetPt
         LDR  R2,[R1]
         CMP  R2,R0
         BNE  NotEmpty
         MOV  R0,#0
         B    done
NotEmpty LDRSB R3,[R2]
         LDR  R4,[SP,#pt]
         STRB R3,[R4]
         ADD  R2,R2,#1
         LDR  R5,=Fifo+FIFOSIZE
         CMP  R2,R5
         BNE  NoWrap
         LDR  R2,=Fifo
NoWrap   STR  R2,[R1]
done     POP  {R4,R5}
         ADD  SP,SP,#4
         BX   LR
```

```
#define FIFOSIZE 10
char volatile *PutPt;
char volatile *GetPt;
char static Fifo[FIFOSIZE];
int Fifo_Get(char *pt){
  if(PutPt == GetPt){
    return(0);
  }
  *pt = *(GetPt);
  GetPt++;
  if(GetPt== &Fifo[FIFOSIZE]){
     GetPt = &Fifo[0];
  }
  return(1);
}
```