Find the contents of the port after each of the following operations:

a) PORTB = 0x65 & 0x76;
b) PORTD = 0x6A ^ 0x6E;
c) PORTB = 0x70 | 0x6B;

a) 0110 0101
   0111 0110
   0110 0100 = PORTB

b) 0110 1010
   0110 1110
   0000 0100 = PORTD

c) 0111 0000
   0110 1011
   0111 1011 = PORTB

Find the port value after each of the following instructions is executed:

a) PORTB = 0x17>>3; b) PORTB = 0x17<<4;

c) PORTB = 1<<5;

a) 0001 0111 => 0000 0010 = PORTB
b) 0001 0111 => 0111 0000 = PORTB
c) 0010 0000 = PORTB

Write a C program using Timer0 without interrupts to generate a 1 kHz square wave on PORT B, bit 3 (PB3) while at the same time transferring data from PORTD to PORTA. Assumer fosc = 8 MHz.

```
// clk/256 => n = 500 us/[(1cc/1MC)(256/8 10^6 cc)] = 15.6
#include <avr/io.h>


int main(void)
{
        DDRB = 0xff;
        DDRD = 0x00;
        PORTD = 0xff;
        DDRA = 0xff;

        // 4000

        TCCR0 = 0x04;
        TCNT0 = -16;

        while(1)
        {

                while (!(TIFR & (1<<TOV0))) {   //0000 0001
```

```
                    PORTA = PIND;
                }

                TCCR0 = 0x00; // turn off timer 0 for reset

                TIFR = 0x01;
                TCNT0 = -16;
                PORTB ^= 0x08;
                TCCR0 = 0x04;

        }

        return 0;
}
```

Using Timer0, write a program in C that toggles pin PORTB.2 every 100 ms for a period of 500 ms. When the timer is not being used, transfer data from PORTD to PORTC. XTAL = 8 MHz. Do not use interrupts for this problem. Create a Microchip Studio workspace for the C language (not C++) (new project => C/C++ => gcc executable project) for the ATmega32, copy your C code, compile the code, and view the disassembly window to see how your C code was translated to assembly language instructions by the compiler. Print out the disassembly window to submit with this problem. Comment on the differences between how you would write the program in assembly and the version generated by the compiler.

Timer 0

100 ms = 100(10^-3) = 100000(10^-6)

n = td/tMC = 100000(10^-6) s/((1 cc/1MC)(1s/8(10^6)cc)) = 800000 MC

```
#include <avr/io.h>

#define TOTALITERATIONS 25
#define TOTALHALFPERIODS 5

Void timerdelay();
Void delayroutine();

unsigned char numberIterations;
unsigned char numberHalfPeriods;


void main(void)
{
        DDRB = 0xff;
        DDRD  = 0x00;
        PORTD = 0xff;
        DDRC = 0xff;

        numberIterations = 0;
        numberHalfPeriods = 0;


  while (1)
        {
                PORTC = PIND;
                delayroutine();
        }
}

void delayroutine()
{

        if (numberIterations < TOTALITERATIONS)
        {
                timerdelay();
        }
        else if (numberHalfPeriods < TOTALHALFPERIODS)
        {
                PORTB ^= 0x04;
                numberIterations = 0;
                timerdelay();
                numberHalfPeriods += 1;
        }
        else
        {
                numberIterations = 0;
                numberHalfPeriods = 0;
                timerdelay();
        }
        numberIterations += 1;
}


void timerdelay()
{
        TCNT0 = -125;
```

```
            TCCR0 = 0x04;
            While (!(TIFR & 1<<TOV0));
            TCCR0 = 0x00;
            TIFR = 1<<TOV0;
}
```

Timer 2

100 ms = 100(10^-3) = 100000(10^-6)

n = td/tMC = 100000(10^-6) s/((1 cc/1MC)(1s/8(10^6)cc)) = 800000 MC

```
#include <avr/io.h>

#define TOTALITERATIONS 25
#define TOTALHALFPERIODS 5

Void timerdelay();
Void delayroutine();

unsigned char numberIterations;
unsigned char numberHalfPeriods;


void main(void)

            DDRB = 0xff;
            DDRD  = 0x00;
            PORTD = 0xff;
            DDRC = 0xff;

            numberIterations = 0;
            numberHalfPeriods = 0;


   while (1)
            {
                        PORTC = PIND;
                        delayroutine();
            }
}

void delayroutine()
{

            if (numberIterations < TOTALITERATIONS)
            {
                        timerdelay();
```

```
	}
	else if (numberHalfPeriods < TOTALHALFPERIODS)
	{
		PORTB ^= 0x04;
		numberIterations = 0;
		timerdelay();
		numberHalfPeriods += 1;
	}
	else
	{
		numberIterations = 0;
		numberHalfPeriods = 0;
		timerdelay();
	}
	numberIterations += 1;
}


void timerdelay()
{
	TCNT2 = -125;
	TCCR2 = 0x06;
	While (!(TIFR & 1<<TOV2));
	TCCR2 = 0x00;
	TIFR = 1<<TOV2;
}
```

Write an assembly program to generate a time delay of 20 us using Timer 0 with Compare Match programming (CTC mode). Assume fosc = 8 MHz.

```
.ORG 0
LDI R16, HIGH(RAMEND)
OUT SPH, R16
LDI R16, LOW(RAMEND)
OUT SPL, R16

SBI DDRB, 2
LDI R17, 0x00
LDI R16, 0x04

LDI R20, 0
OUT TCNT0, R20
LDI R20, 159
OUT OCR0, R20
```

```
MAIN:
OUT PORTB, R17
RCALL DELAY
EOR R17, R16
RJMP MAIN

.ORG 0x100
DELAY:
LDI R20, 0X09
OUT TCCR0, R20
AGAIN: IN R20, TIFR
SBRS R20, OCF0
RJMP AGAIN
LDI R20, 0X00
OUT TCCR0, R20
LDI R20, 1<<OCF0
OUT TIFR, R20
RET
```

In C, program Timer0 to be an event counter (endless loop). Output the current count for each loop to PORTC. Initialize the count to 10. Assume that the external count events are falling edge triggered (see the last 3 bits of TCCR0 to configure as a counter, falling edge).

```
.ORG 0

LDI R20, 0xff;
OUT DDRC, R20

CBI DDRB, 2; PB2 input
SBI PORTB, 2; PULL UP RESISTOR for T0

LDI TCNT0, 10; INITIAL COUNT
LDI R20, 0x06; falling-edge counter
OUT TCCR0, R20


LOOP:
OUT PORTC, TCNT0
RJMP LOOP



#include <avr/io.h>

void main()
{
```

```
DDRC = 0xff;
DDRB &= 0b11111011;
PORTB |= 0b00000100;
TCNT0 = 10;
TCCR0 = 0x06;

While(1)
{
PORTC = TCNT0;
If (TIFR & 1<<TOV0)
{
TCNT0 = 10;
TIFR = 1<<TOV0;
                        }
}
}
```