American International University – Bangladesh
Faculty of Engineering
Department of Electrical and Electronic Engineering

Course Name:	Microprocessor and Embedded Systems with Lab	**Course Code:**	EEE 4103
Semester:	Summer 2021-2022	**Section:**	M
Faculty Name:	**Prof. Dr. Engr. Muhibul Haque Bhuyan**		

Assignment No:	3 **(individual submission consisting of 30 marks)**

Student Name:	**Mostakim Bin Mokbul Rhythm**		
Student ID:	**19-40260-1**	**Program Name:**	**BSc in EEE**

Submission Link (MS forms):			
Submission Date:		**Due Date:**	**10/08/2022**

Special Instruction: Question may be copied from here through copy-paste. Online submission via TEAMS is also allowed. Hardcopy should be submitted to-

Prof. Dr. Engr. Muhibul Haque Bhuyan
Professor
Department of Electrical and Electronic Engineering
(Room # 514, Block D, North Wing)
American International University Bangladesh (AIUB)
408/1 Kuratoli, Dhaka 1229, Bangladesh
Email: muhibulhb@aiub.edu

<u>Questions:</u>

1. Show the set of operations obtained by a parallel adder for a different set of inputs at the function select pins (S_1 S_0) and carry in (C_{in}) at the input pin.
2. Design an adder/subtractor circuit with one selection variable ‘S’ and two inputs ‘A’ and ‘B’: when S = 0 the circuit performs A+B. When S = 1 the circuit performs A-B by taking the 2’s complement of B.
3. Draw the combined logic and arithmetic circuit of ALU where the output can be controlled by changing the value of the mode select pin. List the logic operations that can be performed by a logic circuit based on the value of the input and select pins. Discuss the process to obtain an AND operation from an Equivalence operation with logic expression and diagram.
4. Design an 8-bit shifter circuit for the listed shift functions provided in Table 1. Explain its operation for various cases of select inputs.

Table 1: Functions of control variables

Binary Code	Functions of selection variables					
	A	**B**	**D**	**F with C_{in} = 0**	**F with C_{in} = 1**	**H**
0 0 0	Input Data	Input Data	None	A-1	A	1’s to the output Bus
0 0 1	R1	R1	R1	A+B	A+B+1	Shift Left with I_L=0
0 1 0	R2	R2	R2	A-B-1	A-B	No Shift
0 1 1	R3	R3	R3	A	A+1	Circulate Left with Carry
1 0 0	R4	R4	R4	$\bar{A}$	X	0’s to the output Bus

Binary Code	**Functions of selection variables**					
	A	**B**	**D**	**F with C_{in} = 0**	**F with C_{in} = 1**	**H**
1 0 1	R5	R5	R5	AX**OR** B	X	-
1 1 0	R6	R6	R6	A **AND** B	X	Circulate-Right with Carry
1 1 1	R7	R7	R7	A **OR** B	X	Shift Right with I_R=0

5. Develop the control words in binary and hexadecimal formats using the information provided in Table 1 for the following micro-operations:

i.	R7←R3+R4	ii.	R4←3(R4 – 0)/3
iii.	R3←SHL R3	iv.	Output←R5
v.	R5←R1	vi.	R2←0
vii.	R6←Input	viii.	R6←R4-R2
ix.	R2←SHR R5	x.	R3←CRC R7

One example is shown as follows:

Micro-operation	A	B	D	F	C_{in}	H	In Hex
R5← CRC (R3+R4)	011	100	101	001	0	110	7296h

The necessary bits for the control word are presented in Table 2.

Table 2: 16-bit control word sequence

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A			B			D			F			C_{in}	H		

6. Design a 4-bit ALU for operations listed in Table 1.
7. Develop a flow chart to perform addition and subtraction operations between two signed numbers A and B, where the conditions are A>B and A<B. Write down the microinstruction for each step of the flow chart.
8. Draw the control state diagram to represent the sequences of the developed flow chart of question no. 7.
9. Develop the control memory outputs for the sequence in Table 3 using the information listed in Table 1. To complete the memory outputs, use the microinstructions that you have developed in question no. 7.

Table 3: Control memory bit sequence

			ROM outputs													
ROM Address			Control Word									Address			Mux Select	
			1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0														
0	0	1														
0	1	0														
0	1	1														
1	0	0														
1	0	1														
1	1	0														
1	1	1														

10. Show the flow chart and develop the micro-instruction table for counting the number of 1's in register, R3, and storing the count in register R7. Assume that the starting address is 9. The operation should follow the control word format of Tables 1 and 2. If register, R3 contains the data 10011011 then what would be the contents of the register, R7 after this micro-operation is completed?

    Control memory location contents' format:

Table 4: Control memory bit sequence

						ROM outputs																									
ROM Address						A			B			D			F			C_{in}	H			Mux Select				Address					
						1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
0	0	0	0	0	0																										
0	0	0	0	0	1																										
0	0	0	0	1	0																										
0	0	0	0	1	1																										
0	0	0	1	0	0																										
0	0	0	1	0	1																										
0	0	0	1	1	0																										
0	0	0	1	1	1																										
0	0	1	0	0	0																										
0	0	1	0	0	1																										
0	0	1	0	1	0																										
0	0	1	0	1	1																										
0	0	1	1	0	0																										
0	0	1	1	0	1																										
0	0	1	1	1	0																										
0	0	1	1	1	1																										
0	1	0	0	0	0																										
0	1	0	0	0	1																										
0	1	0	0	1	0																										
0	1	0	0	1	1																										
0	1	0	1	0	0																										
0	1	0	1	0	1																										
0	1	0	1	1	0																										
0	1	0	1	1	1																										

11. Find the baud rate for the three operating modes when f_{OSC} = 8 MHz and UBRRn = 24. Calculate the baud error and comment whether there will be any communication error or not.
12. Calculate the PWM frequency for the output when using fast PWM mode and phase correct PWM when f_{OCO} is 8 MHz and the pre-scale factors are 1, 8, 64, 256, or 1024. Comment on the results afterward.
13. What values should be set in the TCCR0A and TCCR0B registers to operate them in the inverting and non-inverting modes while up or down counting? Which flags are set or reset when counting is completed in the fast and phase correct PWM modes? When the Output Compare (OC0x) is cleared?
14. What is the travel time as per the following program codes of Arduino if the object is at 56 cm from the sonar sensor? Show calculations. How many LEDs are turned ON? If we want that the system will display a message for this distance that how many LEDs are turned ON/OFF (for example, "Two LEDs are turned ON and One LED is turned OFF") then what changes should we include here?

Program code:

```
int distanceThreshold;
int dcm;

long readUltrasonicDistance(int triggerPin, int echoPin)
{
pinMode(triggerPin, OUTPUT); // Clear the trigger
digitalWrite(triggerPin, LOW);
delayMicroseconds(2);

// Sets the trigger pin to HIGH state for 10 microseconds
digitalWrite(triggerPin, HIGH);
delayMicroseconds(10);
digitalWrite(triggerPin, LOW);
pinMode(echoPin, INPUT);

// Reads the echo pin, and returns the sound wave travel time in microseconds
return pulseIn(echoPin, HIGH);
}

void setup()
{
serial.begin(9600);
pinMode(2, OUTPUT);
pinMode(3, OUTPUT);
pinMode(4, OUTPUT);
}

void loop()
{
// set threshold distance to activate LEDs
distanceThreshold = 80;
// measure the ping time and find the distance in cm
dcm = 0.01723*readUltrasonicDistance(7, 6);

serial.print(dcm);
serial.println(" cm");

if (dcm > distanceThreshold) {
        digitalWrite(2, LOW);
        digitalWrite(3, LOW);
        digitalWrite(4, LOW);
        }

if (dcm < distanceThreshold && dcm > distanceThreshold-30) {
        digitalWrite(2, HIGH);
        digitalWrite(3, LOW);
        digitalWrite(4, LOW);
        }
```

```
 if (dcm < distanceThreshold-30 && dcm > distanceThreshold-60) {
        digitalWrite(2, HIGH);
        digitalWrite(3, HIGH);
        digitalWrite(4, LOW);
        }

if (dcm < distanceThreshold-60 && dcm > distanceThreshold-80) {
        digitalWrite(2, HIGH);
        digitalWrite(3, HIGH);
        digitalWrite(4, HIGH);
        }
        delay(100); // Wait for 100 millisecond(s)
        }
```

15. What are the contents of the various registers and various buses and flags after the next clock of Fig. 1 as per Table 4 of question 10 when the ROM addresses are 001011, 001100, 001101, and 001110?

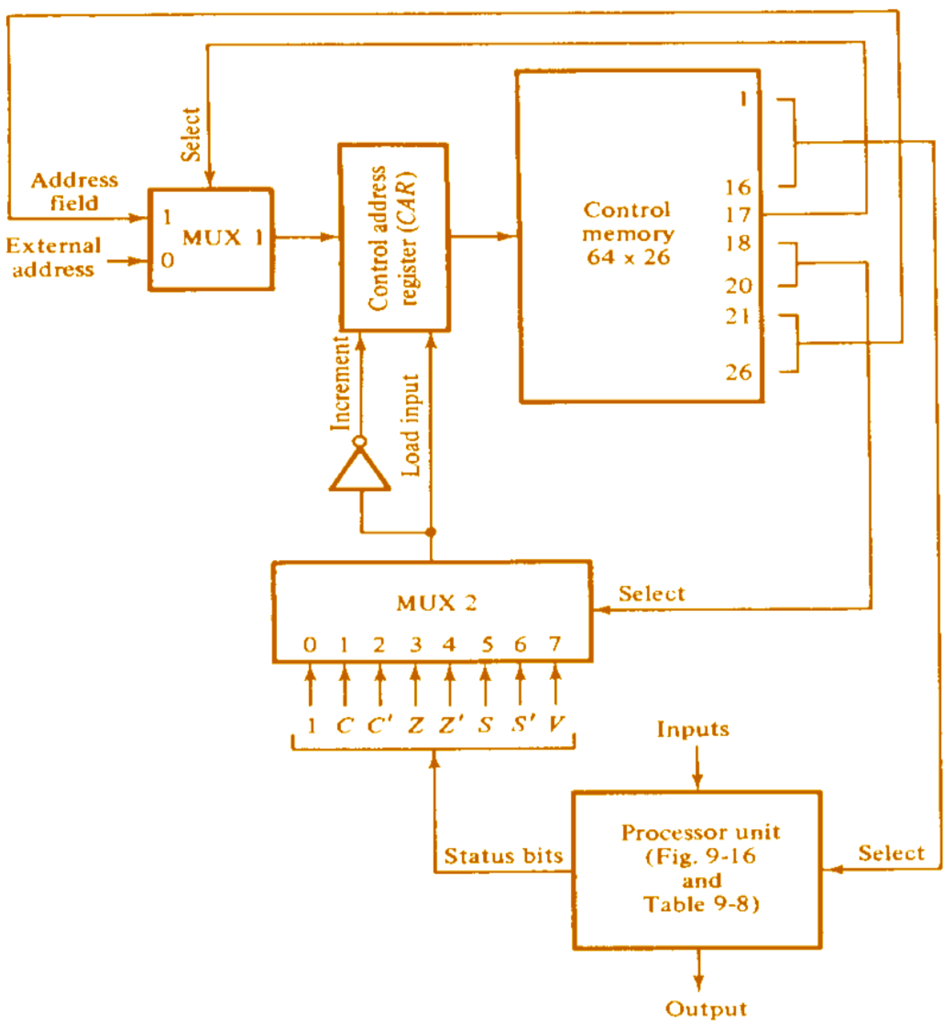


Fig. 1 Micro-program control for processor unit

Mostakim Bin Mokbul Rhythm
19-40260-1

1. Parallel ~~and~~ adder for different set of inputs. Select pins (S1, S0) carry in (Cin)

Input				Output			Decimal sum
A_1	A_0	B_1	B_0	C_0	S_1	S_0	
0	0	0	0	0	0	0	0
0	0	0	1	0	0	1	1
0	0	1	0	0	1	0	2
0	0	1	1	0	1	1	3
0	1	0	0	0	0	1	1
0	1	0	1	0	1	0	2
0	1	1	0	0	1	1	3
0	1	1	1	1	0	0	4
1	0	0	0	0	1	0	2
1	0	0	1	0	1	1	3
1	0	1	0	1	0	0	4
1	0	1	1	1	0	1	5
1	1	0	0	0	1	1	3
1	1	0	1	1	0	0	4
1	1	1	0	1	0	1	5
1	1	1	1	1	1	0	6

19-40260-1

(2) Given,

$S = 0 \longrightarrow A + B$

$S = 1 \longrightarrow A - B$

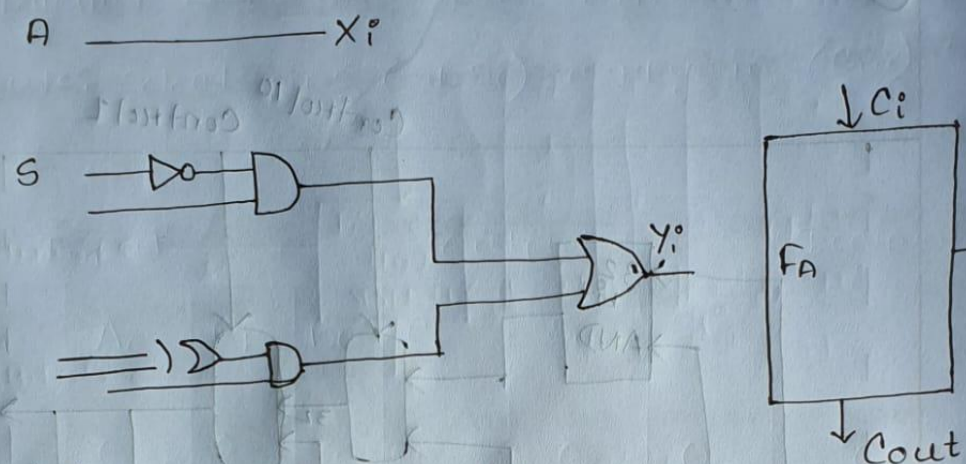
Truth table

S	A_1	B_i	X_i	Y_i
0	0	0	0	0
0	0	1	0	1
0	1	0	1	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0

$X_i = \bar{S}A + SA = A$

$Y_i = \bar{S}B + \bar{B}S$

19-40260-1



if $S = 0/1$ $X_i = A_i$

if $S = 0$ $Y_i = B_i$

$S = 1$ Y_i = 2's complement of B_i

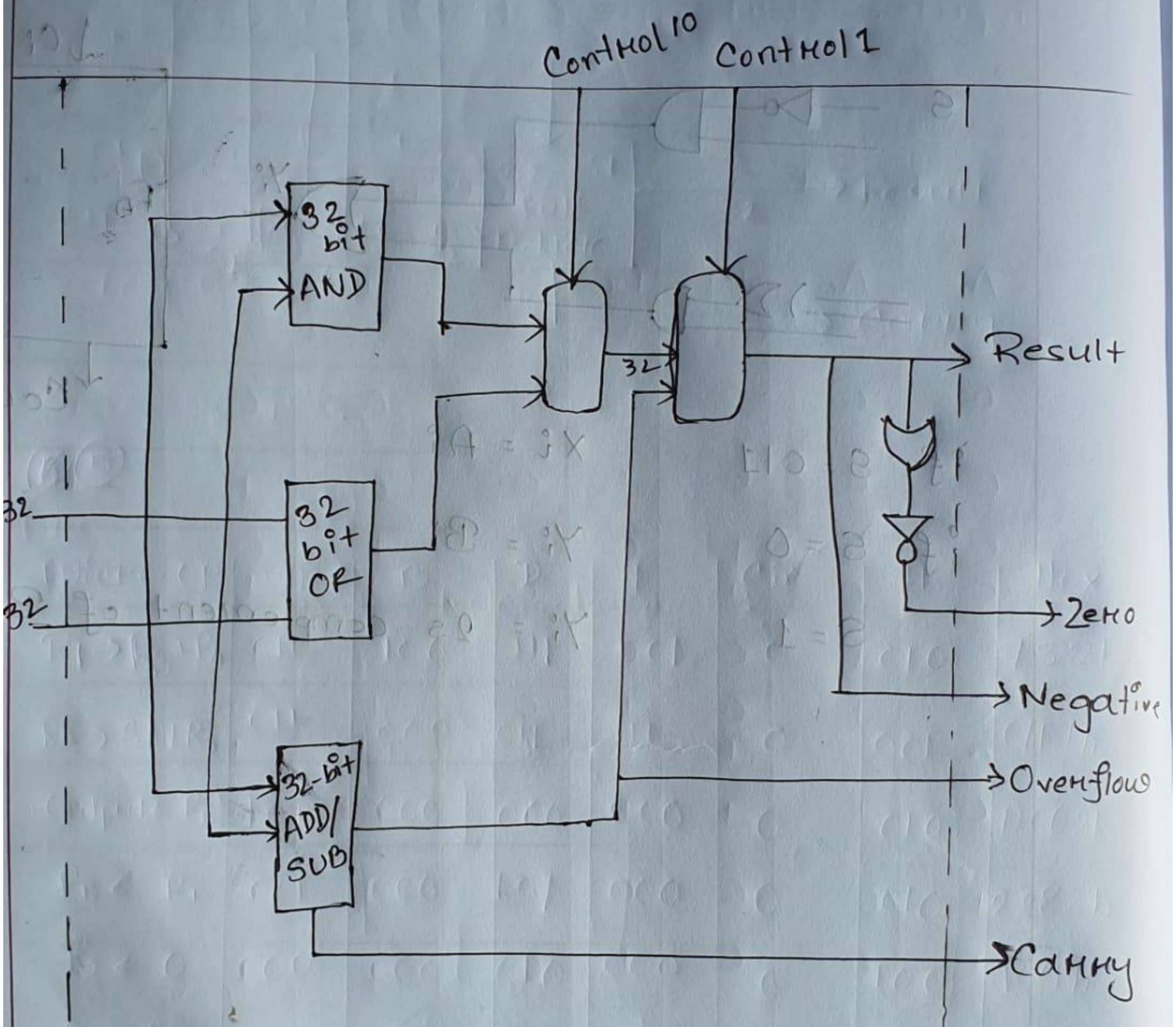
American International University – Bangladesh

Faculty of Engineering

Department of Electrical and Electronic Engineering

19-40260-1

(3)



Real ALU ofcourse use a great deal more operations and performance optimization. This is a just a brief introduction to ALU design. ALU is nothing more than a collection of logic components.

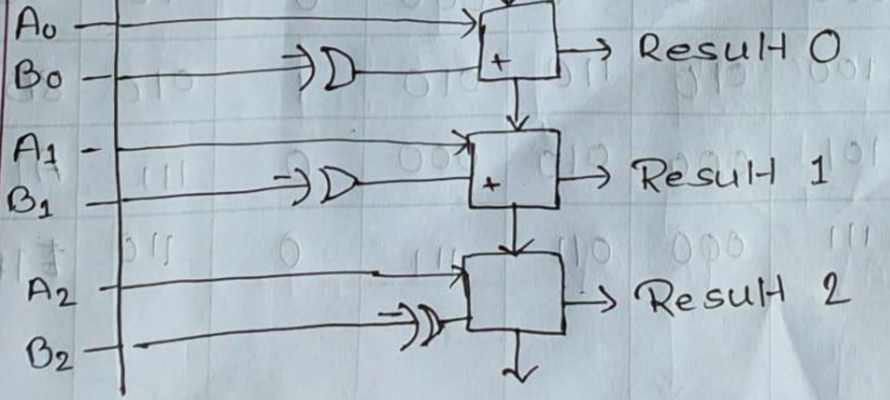
The logic components can be made from standard logic gates data can be flowed currently to multiple units.

Mathematical formula states that $3-2=3+(-2)$ 32 bit complete adder and taken negative one of the input.

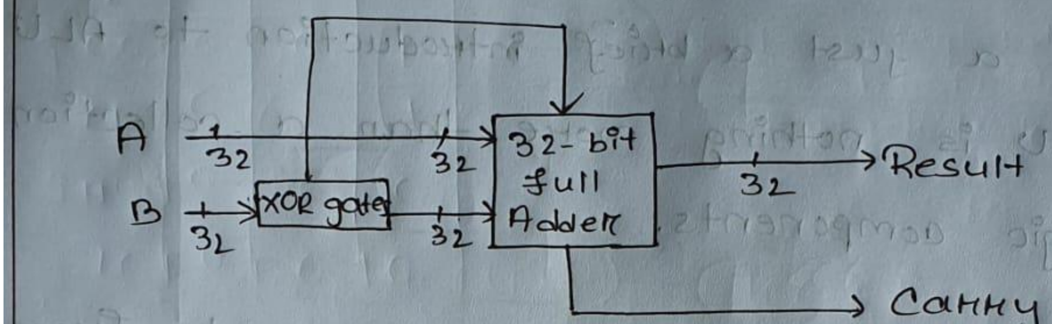
Combining the mentioned state:

$A-B = A+(-B) = A+\sim B+1$

XOR and NOT gate using:



Addition and Substraction:



(5)

Micro-OP	A	B	D	F	Cin	H	Hex
$R_7 \rightarrow R_3+R_4$	011	100	111	001	0	010	7392h
$R_3 \rightarrow SHL R_3$	011	000	011	111	0	001	61F1h
Output ← R_5	101	000	000	000	0	010	A002h
$R_5 \leftarrow R_1$	001	000	101	000	0	010	2282h
$R_2 \leftarrow 0$	000	000	010	000	0	100	104h
$R_6 \leftarrow$ input	000	000	110	000	0	010	302h
$R_6 \leftarrow R_4 - R_2$	100	010	110	010	1	010	8B2Ah
$R_2 \leftarrow SHR R_5$	101	000	010	000	0	111	A107h
$R_3 \leftarrow CRC R_4$	111	000	011	111	0	110	E1F6

(6)

Selection:

S_2	S_1	S_0	C_{in}	Output	X_i	Y_i	Z_i
0	0	0	0	$F = A - 1$	A	1	0
0	0	0	1	$F = A$	A	0	0
0	0	1	0	$F = A + B'$	A	B	0
0	0	1	1	$F = A + B + 1$	A	B	1
0	1	0	0	$F = A - B - 1$	A	B'	0
0	1	0	1	$F = A - B$	A	0	1
0	1	1	0	$F = A$	A	0	0
0	1	1	1	$F = A + 1$	A	1	0
1	0	0	x	$F = \bar{A}$	A	B	0
1	0	1	x	$F = A \oplus B'$	A	B'	0
1	1	0	x	$F = A \cap B$	$A+B'$	0	
1	1	1	x	$F = A \vee B$	$A+B$		

For X_i,

S_2S_1 \ S_0	0	1
00	A	A
01	A	A
11	$A+B'$	$A+B$
10	A	A

For Y_i,

S_2S_1 \ S_0	0	1
00	1	B
01	B'	0
11	B'	0
10	1	B

19-40260-1

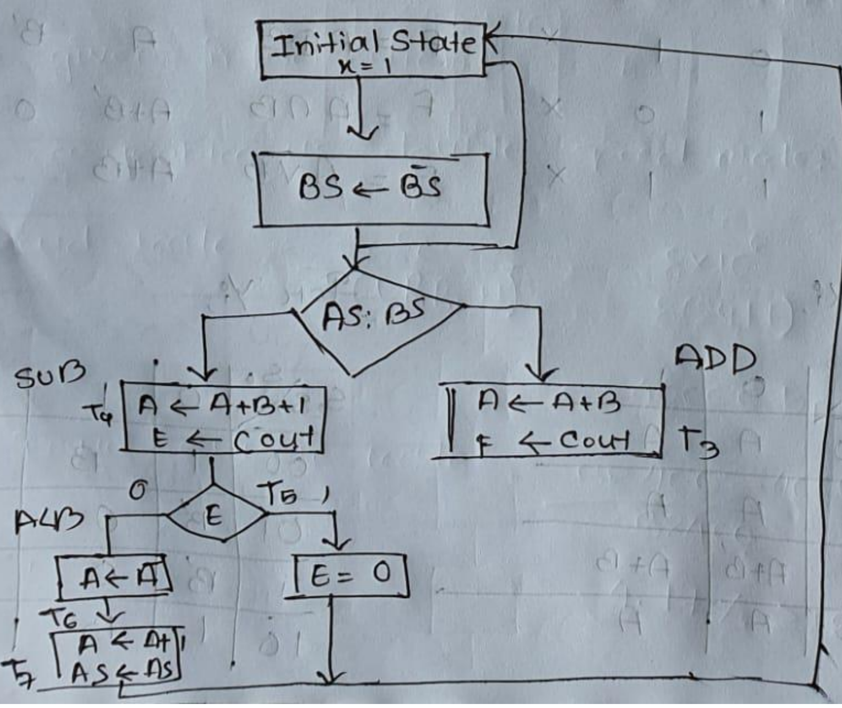
For Z_i,

S_2S_1 \ S_0C_{in}	00	01	11	10
00	0	0	1	0
01	0	1	1	
11	0	0	0	
10	0	0	0	

$$Z_i = S_2' S_1 C_{in} + S_2' S_0 C_{in}$$

7

Flowchart and State diagram:



19-40260-1

(8)

Control State Diagram:

T_0 : Initial State $x=1$

T_1 : $BS \leftarrow \overline{BS}$

T_2 : nothing

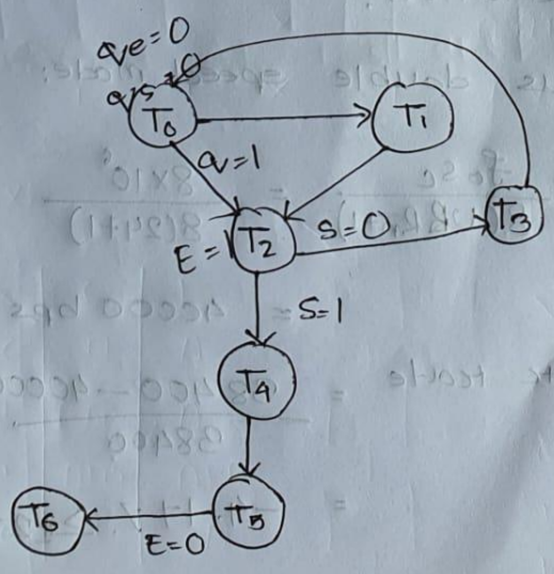
T_3 : $A \leftarrow A+B$, $E \leftarrow Cout$

T_4 : $A \leftarrow A+\overline{B}+1$, $E \leftarrow Cout$

T_5 : $E \leftarrow 0$

T_6 : $A \leftarrow \overline{A}$

T_7 : $A \leftarrow A+1$, $A_s + \overline{A_s}$



American International University – Bangladesh

Faculty of Engineering

Department of Electrical and Electronic Engineering

19-40260-1

(ii)

For asynchronus normal mode:

We know,

$$\text{Baud error rate} = \frac{\text{Standard rate} - \text{Calculated bound rate}}{\text{Standard bound rate}} \times 100$$

$$\text{Baud rate} = \frac{f_{osc}}{16(UBR_n+1)} = \frac{8\times10^6}{16(24+1)}$$

$$= 20000 \text{ bps}$$

$$\text{So, Baud error rate} = \frac{19200-20000}{19200} \times 100\%$$

$$= 4.14\% \geq +2\%$$

So, there will be communication error for the given information.

For asynchronus double speed mode:

$$\text{Baud rate} = \frac{f_{osc}}{8(UBR_n+1)} = \frac{8\times10^6}{8(24+1)}$$

$$= 40000 \text{ bps}$$

$$\text{So, Baud error rate} = \frac{38400-40000}{38400} \times 100\%$$

$$= -4.17\% \geq \pm2\%$$

So, there will be communication error for the given information.

∴ For synchronus master mode:

$$\text{Baud rate} = \frac{f_{osc}}{2(U_{B}RR_n+1)}$$

$$= \frac{8\times10^6}{2(24+1)}$$

$$= 160000 \text{ bps}$$

So, Baud error rate $= \frac{128000-160000}{128000} \times 100\%$

$$= -25\% \geq \pm 2\%$$

So, there will be communication error for the given information.

(12)

The PWM frequency for the post PWM mode $f_{OCn} \times PWM = \frac{f_{clk}\ 10}{N \times 256} = \frac{10}{1024 \times 256}$

$= 38.10$

The PWM frequency for the Phase correct PWM mode:

$$f_{OCn} \times P_{CPWM} = \frac{f_{clk}\ 10}{N \times 510}$$

$$= \frac{10}{1024 \times 510}$$

$$= 19.15\ Hz$$

19-40260-1

(15)

The register is given serial input with serial data 101101, the first digit from the left is replaced with the first digit from the right of serial data.

The contents of the register after 1st pulse 1110.
Similarly, after the second shift, the contents of the register are 0111.

After the 3rd clock pulse, we get 1011.
After the 4th clock pulse, we get ~~10~~1101.
After the 5th clock pulse, we get 0110
After the 6th clock pulse, we get 1~~0~~011

19-40260-1

(13)

In the TCCR0A and TCCR0B registers the COM0x1:0 bits to 10 values should be set to operate them in the inverting and the COM0x1:0 bits to 11 to operate in the non-inverting modes.

TCCR0A

Bit 0x24(0x46)	COM0A1	COM0A0	COM0B1	3	2	WGM01	WGM00
Read-write	R/W	R/W	R/W	R	R	R/W	R/W
Initial value	0	0	0	0	0	0	0

TCCR0B

Bit 0x 25(0x45)	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00
	W	W	R	R	R/W	R/W	R/W	R/W
Initial value	0	0	0	0	0	0	0	0

OCRnx and TOV_n interrupt flag need to be set or reset while counting is finished in the rapid and accurate PWM modes. The timer overflow flag (TOV0) is ready whenever the counter reaches bottom. The incurrept Flag can be used generate every ocassion in the counter reaches the lowest fee.

The PWM waveform is generated through OCRx sign up examine suit betⁿ OCRx & TCNTO where the counter increments and setting the OCRx check in.

4.

Binary Code	**Functions of selection variables**					
	A	**B**	**D**	**F with C_{in} = 0**	**F with C_{in} = 1**	**H**
0 0 0	Input Data	Input Data	None	A-1	A	1’s to the output Bus
0 0 1	R1	R1	R1	A+B	A+B+1	Shift Left with I_L=0
0 1 0	R2	R2	R2	A-B-1	A-B	No Shift
0 1 1	R3	R3	R3	A	A+1	Circulate Left with Carry
1 0 0	R4	R4	R4	$\bar{A}$	X	0’s to the output Bus
1 0 1	R5	R5	R5	AX**OR** B	X	-
1 1 0	R6	R6	R6	A **AND** B	X	Circulate-Right with Carry
1 1 1	R7	R7	R7	A **OR** B	X	Shift Right with I_R=0

9.

ROM Address			ROM outputs													
			Control Word									Address			Mux Select	
			1	2	3	4	5	6	7	8	9	10	11	12	13	14
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0	0	0	1	0	0	1
0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0
0	1	1	0	0	0	1	0	1	0	0	0	0	0	0	0	1
1	0	0	0	0	1	0	1	1	0	0	0	1	0	1	0	1
1	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1
1	1	0	0	1	1	1	0	1	0	0	0	1	1	1	0	1
1	1	1	0	0	0	0	1	1	0	1	0	0	0	0	0	1

14.

Step1 | Setting up the LEDs

Open up Arduino IDE and open a new sketch by going to File>New. Let's get the LEDs working first. Go ahead and delete the default

So that you get a clear screen.This int is for the brightness of the LED. This number can range anywhere between 0 and 255 with 0 being completely off and 255 being completely on. For example

Now let's test each LED to make sure there working. In between the voidsetup() and the color functions write a voidloop() function. voidloop() executes the code inside it then loops to the beginning of the function and does it all over again. voidloop() is were all the "Moving pieces" of your code goes. In the voidloop() function set all of the LEDs to completely on.

```
void loop(){
   Red(255);
   Green(255);
```

```
  Blue(255);
}
```

Now we need to define our pins the red pin is connected to pin 3. The green pin is connected to pin 4. And the blue pin is connected to pin 5. So lets make 3 contsint with values as 3, 4, and 5.

By now your script should look like this:

```
void Blue(int VAL){
 analogWrite(redPin, 0);
 analogWrite(greenPin, 0);
 analogWrite(bluePin, VAL);
}
const int redPin = 3;
const int greenPin = 4;
const int bluePin = 5;
```

conclusion:

All three LEDs should now light up. If not check fig1 and make sure your wiring is correct. Then check for errors in your code. If all else fails see the Questions or Comments at the bottom of the tutorial.

If all LEDs light up. Congratulations! You have completed step one of the tutorial. Move on the Step 2.

Step2

Now that we have the LED lights working we need to turn our attention to the HC-SR04 sonar sensor. But first, if you haven't already, let's turn off the LEDs by removing the color functions form the voidloop() like this.

```
void loop(){
//Nothing to see here :-]
}
```

First at the top of our sketch we need to set our pins for echo and trigger.

```
const int echoPin = 9;
const int trigPin = 8;
```

We also need to make another int to keep track of our distance. Do not set this as const because this value will change. To stay organized put this in-between the pin int and the void setup()

```
int distance;
```

NOTE: Do not set int distance to equal anything yet. We will do that later.

Next we need to set echoPin as INPUT and trigPin as OUTPUT inside of voidsetup()

```
void setup(){
 pinMode(redPin, OUTPUT);   /*
 pinMode(greenPin, OUTPUT);    This code should already be here
 pinMode(bluePin, OUTPUT);   */
 pinMode(echoPin, INPUT);    // Add this line
```

```
 pinMode(trigPin, OUTPUT);   // Add this line
}
```

Now we need to code the sonic burst. To save space I'm going to call it a ping from now on. So first we set trigPin to LOW then to HIGH for 10 µs then back to LOW. To stay organized lets do that in a void. How do you set a pin to LOW or HIGH? I'm glad you asked. First we need to understand what LOW and HIGH even mean. LOW and HIGH are constant values. When a pin is set as OUTPUT in the pinMode(), LOW means that a pin is at 0V. (on 5V boards). HIGH means that a pin is at 5V. To set a pin as LOW or HIGH we use the digitalWrite(); function digitalWrite(); takes in two values. First it takes in a int value as the pin key. Just like analogWrite(); this value can either be in the form of physical number (like 3 for example) or as a int name (like redPin for example). The second value it takes in is a state value (normally LOW or HIGH). Now that we know how to do it, let's code the ping in a void function that we'll call sendPing();

```
void sendPing(){
 digitalWrite(trigPin, LOW);
 delayMicroseconds(5);
 digitalWrite(trigPin, HIGH);
 delayMicroseconds(10);
 digitalWrite(trigPin, LOW);
}
```

Now in the void loop() we need to call the sendPing() function then we need to figure out how long echoPin reads HIGH. This is easy with the function pulseIn(); which takes a pin key (either as a number or int name) and a state (either HIGH or LOW) then it reads how long said pin key was HIGH or LOW, then returns the value as a int in µs. So let's make a new int and call it duration. duration will be our Time value in the equation above. Now we can set distance to equal to duration x 0.034/2. Code it like this.

Step3

Now it is finally time to take all of our information we have gathered and make it do something! This is actually the simple part of project. All we have to do is write a string of if() statements at the bottom of void loop().

First we need to decide at what distances which light we want to show.

Here are my choices;

- Red @ distances btw 33 - 50 cm
- Green @ distances btw 16 - 32 cm
- Blue @ distances btw 1 - 15 cm

Now we write our if() statement at the bottom of void loop()

```
void loop(){
sendPing();                              /*
int duration = pulseIn(echoPin, HIGH);
distance = 0.034 * duration / 2;           This code should already be here
distance = clamp(distance, 0, 50);
printToScreen(distance);                  */
if(distance >= 0 && distance <= 15){
```

```
  Blue(255);
  Green(0);
  Red(0);
  }
else if(distance >= 16 && distance < 33){       Add these lines of code
  Green(255);
  Blue(0);
  Red(0);
  }
  else if (distance >= 33 && distance <= 50){
   Red(255);
   Green(0);
   Blue(0);
  }                                          */
          }
```