MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

Microcontrollers Lab

Laboratory Manual: ICE 3161

Fifth Semester B.Tech.

Name of the student:.....

Registration Number:.....

Department of Instrumentation and Control Engineering

Manipal Institute of Technology

MANIPAL - 576104

CERTIFICATE

This is to certify that the Laboratory Manual/Journal for the lab titled **Microcontrollers Laboratory (ICE 3161)** submitted by Mr./Ms._____

(Reg. No :_____) of fifth semester, Electronics and Instrumentation Engineering for the academic year _____ , as per laboratory course requirements, which has been evaluated and duly certified.

Place: Manipal

Date: **Lab In-Charge**

CONTENTS

Evaluation Plan:

- **Continuous Evaluation - 50%**
  - *Regularity, Preparation, Documentation, Viva and Performance.*
- **Mini Project - 10%**
- **Lab test - 40%**

1. Data Transfer

1. **Program to transfer 5 bytes of data from internal memory location starting from 20h to internal memory location starting from 40h.**

***Program*:**

LABELS	MNEMONICS	COMMENTS
	ORG 00H	; Start of program
START:	**MOV R2, #05H**	;Initialize the counter
	MOV R0, #20H	;Initialize the source memory ;location to 20h
	MOV R1, #40H	;Initialize the destination memory ;location to 40h
AGAIN:	**MOV A, @R0**	;Copy a byte of data 20h to the ;accumulator
	MOV @R1, A	;Copy content of accumulator to ;40h
	INC R0	;Increment the source memory ;location by 1
	INC R1	;Increment the destination memory ;location by 1
	DJNZ R2, AGAIN	Decrement the counter, ;if count ≠0, jump to AGAIN
	END	; End of program.

Result:

Before Execution:

Source Address	Data
20h	0x01
21h	0x02
22h	0x03
23h	0x04
24h	0x05

Destination Address	Data
40h	0x00
41h	0x00
42h	0x00
43h	0x00
44h	0x00

After execution:

Source Address	Data
20h	0x01
21h	0x02
22h	0x03
23h	0x04
24h	0x05

Destination Address	Data
40h	0x01
41h	0x02
42h	0x03
43h	0x04
44h	0x05

2. **Program to interchange two blocks of data residing at internal memory locations starting from 20h and 40h.**

Program:

LABELS	MNEMONICS	COMMENTS
	ORG 00H	; Start of program
START:	**MOV R2, #05H**	**;** Initialize the counter
	MOV R0, #20H	**;** Initialize the source memory location
	MOV R1, #40H	**;** Initialize destination memory location
AGAIN:	**MOV A, @R0**	
	XCH A, @R1	**;** Exchange A with data pointed to by R0
	MOV @R0, A	**;** Copy the contents of A to the address in ;R0
	INC R0	**;** Increment the source memory pointer
	INC R1	**;** Increment the destination memory pointer.
	DJNZ R2, AGAIN	**;** Decrement the counter, ;if count ≠0, jump to AGAIN
	END	; End of program

Result:

Before Execution:

Source Address	Data
20h	0x01
21h	0x02
22h	0x03
23h	0x04
24h	0x05

Destination Address	Data
40h	0x11
41h	0x12
42h	0x13
43h	0x14
44h	0x15

After execution:

Source Address	Data
20h	0x11
21h	0x12
22h	0x13
23h	0x14
24h	0x15

Destination Address	Data
40h	0x01
41h	0x02
42h	0x03
43h	0x04
44h	0x05

3. **Program to transfer 5 bytes of data from external memory location starting from 9400h to external memory location starting from 9500h.**

Program:

LABELS	MNEMONICS	COMMENTS
	ORG 00H	; Start of program
START:	**MOV R0, #05H**	**;** Initialize the counter.
	MOV DPTR, #9400H	**;**Initialize memory location 9400h
	MOVX A, @DPTR	;Copy the contents of address in ;DPTR to A
	MOV R1, 82H	**;**Save the address in R1 and R2
	MOV R2, 83H	
	MOV DPTR, #9500H	**;**Initialize memory location 9500h
AGAIN:	**MOVX @DPTR, A**	;Copy data from A to the external ;address in DPTR
	INC DPTR	
	PUSH 82H	**;** Push the address onto the stack
	PUSH 83H	
	MOV 82H, R1	**;** Save the address in DPL and DPH
	MOV 83H, R2	
	INC DPTR	; Increment the memory location by 1
	MOVX A, @DPTR	;Copy the data byte in accumulator ;to destination memory location
	MOV R1,82h	
	MOV R2,83h	
	POP 83h	; Rctricvc thc dcstination mcmory ;location address from the stack

	POP 82h	
	DJNZ R0, AGAIN	**;** Decrement the counter, ;if count ≠0, jump to AGAIN
	END	; End of program

Result:

Before Execution:

Source Address	Data
9400h	0x01
9401h	0x02
9402h	0x03
9403h	0x04
9404h	0x05

Destination Address	Data
9500h	0x00
9501h	0x00
9502h	0x00
9503h	0x00
9504h	0x00

After execution:

Source Address	Data
9400h	0x01
9401h	0x02
9402h	0x03
9403h	0x04
9404h	0x05

Destination Address	Data
9500h	0x01
9501h	0x02
9502h	0x03
9503h	0x04
9504h	0x05

4. Program to interchange 5 bytes of data residing at external memory locations starting from 9400h and 9500h.

Program:

LABELS	MNEMONICS	COMMENTS
	ORG 00h	; Start of program
START:	**MOV R0, #05H**	**;** Initialize the counter
	MOV DPTR, #9400H	**;** Initialize memory location 9400h
	MOV R1, 83H	**;** Save the address in R1 and R2**.**
	MOV R2, 82H	
	MOV DPTR, #9500H	**;** Initialize memory location 9500h
AGAIN:	**PUSH 82H**	**;** Push the address onto the stack
	PUSH 83H	
	MOVX A, @DPTR	**;** Copy a byte of data from 9500h to ;accumulator
	MOV R3, A	**;**Save in R3
	MOV 83H, R1	
	MOV 82H, R2	
	MOVX A, @DPTR	**;** Copy a byte of data from 9400h to ;accumulator
	XCH A, R3	**;** Exchange data byte between R3 and ;A
	MOVX @DPTR, A	
	INC DPTR	**;** Increment the memory location ;94xx
	MOV R1, 83H	; Save the address in R1 and R2
	MOV R2, 82H	

	POP 83H	; Retrieve the destination memory ;location address from the stack
	POP 82H	
	MOV A, R3	; Copy the data byte in accumulator to ;destination memory location
	MOVX @DPTR, A	
	INC DPTR	; Increment the memory location 95xx
	DJNZ R0, AGAIN	; Decrement the counter, ;if count ≠0, jump to AGAIN
	END	; End of program

Result:

Before Execution:

Source Address	Data
9400h	0x01
9401h	0x02
9402h	0x03
9403h	0x04
9404h	0x05

Destination Address	Data
9500h	0x11
9501h	0x12
9502h	0x13
9503h	0x14
9504h	0x15

After execution:

Source Address	Data
9400h	0x11
9401h	0x12
9402h	0x13
9403h	0x14
9404h	0x15

Destination Address	Data
9500h	0x01
9501h	0x02
9502h	0x03
9503h	0x04
9504h	0x05

2. Arithmetic Operations

1. Program to add two 16-bit numbers residing at internal memory locations.

Program:

LABELS	MNEMONICS	COMMENTS
	ORG 00h	; Start of program
START:	**CLR C**	; Clear carry
	MOV R2, #02H	; Initialize the counter
	MOV R0, #20H	; Initialize the pointer to first number
	MOV R1, #40H	; Initialize the pointer to second ;number
LOOP:	**MOV A, @R0**	
	ADDC A, @R1	; Perform addition with carry
	MOV @R0, A	; Save the result in address pointed by ;R0
	INC R0	;Increment the pointer by one
	INC R1	
	DJNZ R2, LOOP	**;** Decrement count if count ≠0, jump ;to LOOP
	CLR A	
	ADDC A, #00H	; Perform if there is a carry
	MOV @R0, A	; Storing the number to next location
	END	; End of program

Result:

Input:

Number 1 = 0A85h (16 bit)

Number 2 = 1234h (16 bit)

Address	Data
20h	0x85
21h	0x0A
40h	0x34
41h	0x12

Output:

Addition Result= 1CB9h

Address	Data
20h	0xB9
21h	0x1C

2. **Program to multiply a 16 bit number by an 8 bit number residing at internal memory locations.**

Program:

LABELS	MNEUMONICS	COMMENTS
	ORG 00H	; Start of program
START:	**MOV R0, 30H**	**;** Load the 8 bit number in R0
	MOV A, 31H	**;** Load the lower byte of 16 bit number in A**.**
	MOV B, R0	**;** Move the lower byte of 16 bit number to A
	MUL AB	**;** Perform multiplication
	MOV 50H, A	**;** Store the lower byte result in 50h
	MOV R1, B	**;** Store the higher byte of result in R1
	MOV A, 32H	**;** Load the higher byte of 16 bit ;number to A

	MOV B, 30H	; Move 8 bit number in to B
	MUL AB	; Perform multiplication.
	MOV R2, B	; Store the higher byte of result in R2
	ADD A, R1	; Perform addition
	JNC FINISH	; Check if C=1, If not jump to ;FINISH
	INC R2	; If CY =1, increment higher byte by ;one.
FINISH:	**MOV 51H, A**	; Store the middle byte result in 51h.
	MOV 52H, R2	; Store the higher byte result in 52h.
	END	; End of program

Result:

Input:

Number 1 = 11h (8 bit)

Number 2 = 3322h (16 bit)

Output:

Multiplication Result= 036542h

Source Address	Data
30h	0x11
31h	0x22
32h	0x33

Destination Address	Data
50h	0x42
51h	0x65
52h	0x03

3. Program to find the square root of a number residing at register R3.

Program:

LABELS	MNEMONICS	COMMENTS
	ORG 00H	;Start of program
START:	**MOV R3,# 36**	;Load decimal number 36 in R3
	MOV R0,#00H	;Clear R0
	MOV R1,01H	Initialize with R1=1(First odd number)
LOOP1:	**CLR C**	
	MOV A, R3	;Load a with the number 36
	SUBB A, R1	;Calculate(36-odd number)
	MOV R3, A	;Load (36-odd number) in A
	JNC SQR	;Check if C=1, If not jump to SQR
	MOV A, R0	;If C=1,load square root in A
	MOV P0, A	;Send the square root answer to P0
LOOP:	**SJMP LOOP**	;Loop here forever
SQR:	**INC R0**	;Increment R0 by 1, if 36>odd ;number and C=0
	MOV A, R1	;Load A=R1
	ADD A, #02H	;Calculate next odd number
	MOV R1, A	;Load R1 with next odd number
	SJMP LOOP1	;Repeat the process until C=1
	END	; End of program.

Result:

Input **: Load number 36 in R3.**

Output: **Send result (06) to P0.**

Exercise Program

1. **Analyze the following code and show the results on the registers A and B after each step.**

```
ORG 00H
MOV A, #30H
SJMP 100H
MOV A, #40H
ORG 100H
MOV B, #0CFH
ANL A, B
SJMP 200H
ORG 200H
XRL A, #0FFH
SJMP 200H
```

2. **Write an 8051 ALP to compute the number of zeros and ones in the following 8 bit-stream. 1011 1100**

3. **Write an 8051 ALP to add five 8-bit numbers.**

3. Code Conversion

1. Program to convert Packed BCD to ASCII numbers.

Program:

LABELS	MNEUMONICS	COMMENTS
	ORG 00H	;Start of program
START:	**MOV A, #25H**	;A=25H, packed BCD
	MOV R2, A	;Keep a copy of BCD data IN R2
	ANL A, #0FH	;Mask the upper nibble (A=05)
	ORL A, #30H	;Make an ASCII, A=35II
	MOV R6. A	;Save it in R6
	MOV A, R2	;A=25H, Get the original data
	ANL A, #0F0H	;Mask the lower nibble (A=20)
	RR A	;Rotate right
	RR A	;Rotate right
	RR A	;Rotate right
	RR A	;Rotate right
	ORL A, #30H	;Make an ASCII, A=32H
	MOV R2, A	;Save it in R2
	END	; End of program

Result:

Input:

A=25H

Output:

R2=32H

R6=35H

2. Program to form a 2's complement number given the 7-bit magnitude and sign information separately.

Program:

LABELS	MNEUMONICS	COMMENTS
	ORG 00H	;Start of program
START:	**MOV P0, #00H**	
	MOV R0, #3FH	; Load R0 with 3Fh
	MOV R1, #01H	;Load R1 with sign bit 01h
	MOV A, R1	;Move assign bit to A
	JB ACC.0, CONV	;Check A for sign bit, if 1 branch ;to CONV
DIS:	**MOV A, R0**	;Move the value in R0 to A
	MOV P0, A	;Send 2's complement number to ;P0
STOP:	**SJMP STOP**	
CONV:	**MOV A, R0**	;Move magnitude in A
	CPL A	;Complement A
	INC A	; Add 1 to A
	MOV R0, A	; Save 2's complement number to R0
	SJMP DIS	
	END	; End of program

Result:

Input:
Assume 7-bit magnitude information is available in R0 and sign bit is available in R1.
Output:
Send 2's complement number to Port 0.

3. Program to convert hexadecimal to its equivalent decimal number.

Program:

LABELS	MNEUMONICS	COMMENTS
	ORG 00H	; Start of program
START:	**MOV R0, #30H**	
	MOV A, P1	; Read data from P1
	MOV B, #10	; B=0AH
	DIV AB	; Divide by 10
	MOV @R0, B	; Save lower digit
	INC R0	
	MOV B, #10	
	DIV AB	; Divide by 10 once more
	MOV @R0, B	; Save the next digit
	INC R0	
	MOV @R0, A	; Save the last digit
	END	; End of program

Result:

Input:
Read the data from Port P1 (FEH).

Output:
Converted data is available in internal memory locations starting from 30H.

Exercise Program

1. **Write an 8051 ALP to convert ASCII numbers to packed BCD number.**

2. **Write an 8051 ALP to convert a decimal number to its equivalent hexadecimal number.**

3. **Write an 8051 ALP to separate an 8-bit 2's complement number into magnitude and sign bit.**

4. Array Handling

1. Program to find the smallest number in a given array residing at external memory locations starting from 9400h.

Program:

LABELS	MNEMONICS	COMMENTS
	ORG 00H	; Start of program
START:	**MOV R0, #09H**	; Initialize the counter
	MOV DPTR, #9400H	; Initialize the array
	MOVX A, @DPTR	; Get a byte of data from the array
	MOV R1, A	; Save the number in R1
AGAIN:	**INC DPTR**	; Increment data pointer
	MOVX A, @DPTR	; Get the next number.
	MOV R2, A	; Save it in R2.
	CLR C	; Clear carry flag
	SUBB A, R1	; Subtract the first number from the second number
	JNC OVER	; If Carry = 0 , R1 contains the smallest number
	MOV R1, 02H	; If carry = 1, R2 contains the smallest number
OVER:	**DJNZ R0, AGAIN**	; If count ≠0 jump to AGAIN
	MOV DPTR, #9500H	
	MOV A, R1	
	MOVX @DPTR, A	; Save the result
	END	; End of program

Result:

Before Execution:

Source Address	Data
9400h	0x31
9401h	0xA8
9402h	0x25
9403h	0x12
9404h	0x01
9405h	0x43
9406h	0xE0
9407h	0x17
9408h	0x10
9409h	0x20
9400h	0x31

After Execution:

Destination Address	Data
9500h	0x01

Exercise Program

1. **Write an 8051 ALP to find the largest number in a given array.**
2. **Write an 8051 ALP to sort the given array in ascending order.**
3. **Write an 8051 ALP to sort the given array in descending order.**

5. I/O Programming

1. Program to toggle all bits of Port 1 by sending the values 55h & AAh continuously. Put a time delay of 0.6 ms.

Program:

LABELS	MNEUMONICS	COMMENTS
	ORG 00H	;Start of program
REPEAT:	**MOV P1, #55H**	; Send 55h to Port 1
	LCALL DELAY	; Time delay
	MOV P1, #0AAH	; Send AAh to Port 1
	LCALL DELAY	; Call time delay
	SJMP REPEAT	; Repeat toggling
DELAY:	**MOV R0, #0FFH**	; Routine for 0.6 m sec delay
AGAIN:	**DJNZ R0, AGAIN**	
	RET	
STOP:	**SJMP STOP**	
	END	; End of program

Result:

The output is observed on Port 1.

2. **Program to get the x value from P1 and send x^2 to P2 continuously. Access the x2 values from lookup table located in the ROM space of 8051.**

Program:

LABELS	MNEUMONICS	COMMENTS
	ORG 00H	;Start of program
	MOV DPTR, #XSQR	**;** Load the look-up table ;address
BACK:	**MOV A, P1**	**;** Get x value from Port 1
	MOVC A, @A+DPTR	**;** Get x squared from table
	MOV P2, A	**;** Issue it to Port 2
	SJMP BACK	**;** Keep doing it
	ORG 100H	
XSQR:	**DB 0, 1, 4, 9, 16, 25, 36, 49**	
	DB 64, 81, 100, 121, 144, 169, 196, 225	
	END	; End of program

Result:

The output is observed on Port 2 based on the status of Port 1.

3. **A washing machine is designed for voltage range of 180-240 V. If the voltage is above 240V or below 180V, the washing machine will shut down by turning off a relay connected to P1.0. Assume that the voltage can be read at Port 0 in the range 0-255V. Write a program to implement this operation.**

LABELS	MNEUMONICS	COMMENTS
	ORG 00H	;Start of program
	SJMP 100H	; Jump to location 100H
	ORG 100H	
INPUT:	**MOV P0, #0FFH**	;Configure P0 as input port
	MOV A, P0	;Input the voltage value to A
	SUBB A, #180	:A=A-180
	JC OFF	;If C=1, then A<180
	MOV A, P0	;Input thc voltagc valuc to A
	SUBB A, #240	:A=A-240
	JNC OFF	;If C=0, then A>240
	SJMP INPUT	;Voltage in range check the ;input again
OFF:	**CLR P1.0**	;Turn off the relay
	SJMP INPUT	;Check the input voltage ;again
	END	; End of program

Result:

The output is observed on Port 1.

Exercise Program

1. **Write a program generate an 8 bit BCD up counter counting from 00h to 99h. Store the result in memory location starting from 9400h. Introduce a time delay of 0.8 ms between each count.**

2. **Generate the following waveform on P1.3.**

3. **Generate two square waves – one of 2KHz frequency at pin P1.3 and another of frequency 4KHz at pin P1.5.**

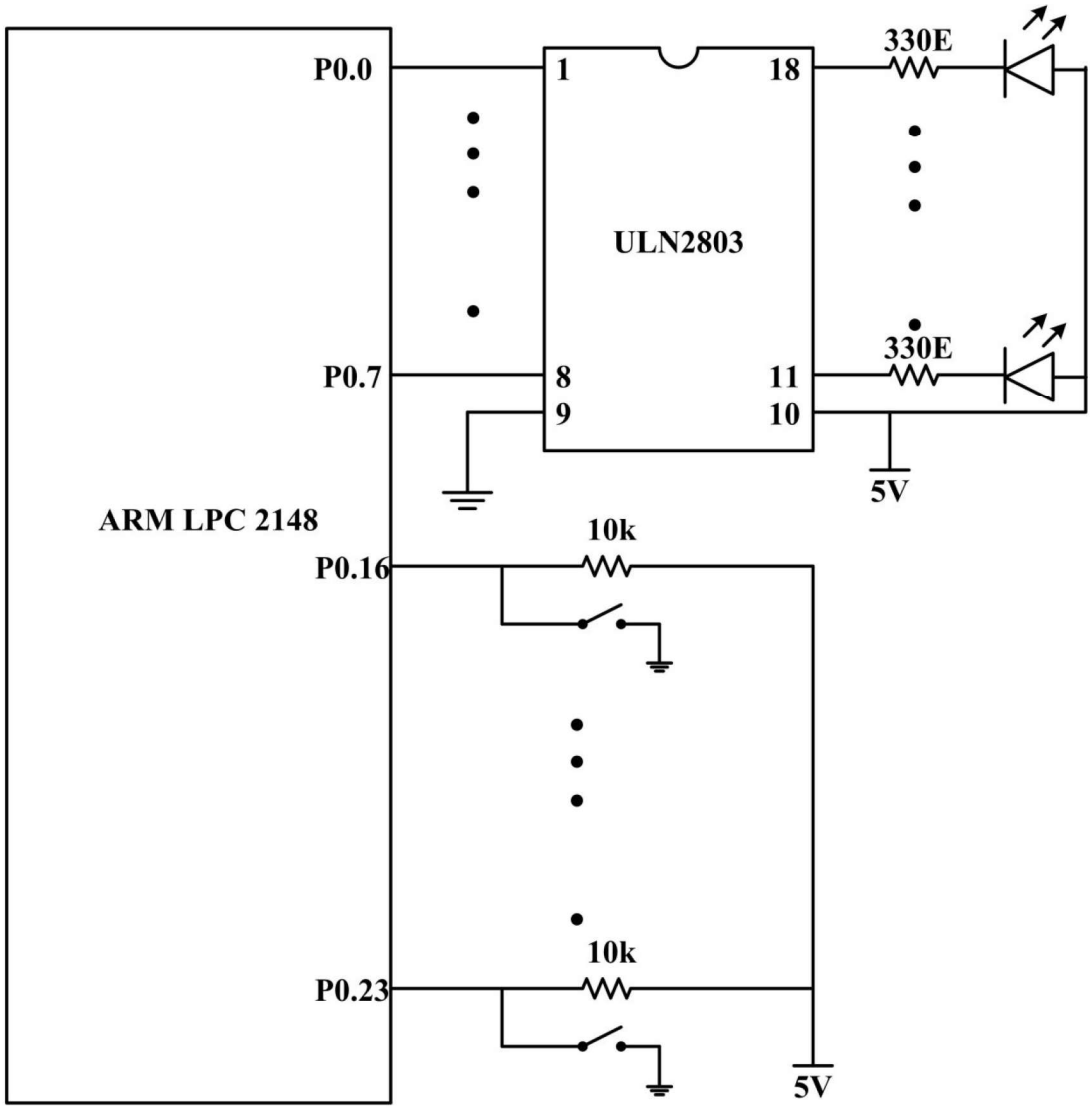
6. LED and Toggle Switch Interface

1. **Program to blink LEDs using ARM LPC 2148.**

Line #	Program	Comments
1	**#include <LPC214X.H>**	
2	**void DELAY(unsigned long VALUE);**	
3	**unsigned char A = 0x01;**	
4	**int main()**	
5	**{**	
6	**IO0DIR = 0x000000FF;**	/* Port0 00-07 as output*/
7	**while(1)**	/* Infinite loop */
8	**{**	
9	**IO0SET |= A;**	/* Port0 00-07 High*/
10	**DELAY(1000000);**	
11	**IO0CLR |= A;**	/* Port0 00-07 Low*/
12	**A <<=1;**	
13	**if(A==0x00) A=0x01;**	
14	**DELAY(1000000);**	
15	**}**	
16	**}**	
17	**void DELAY(unsigned long VALUE)**	
18	**{**	
19	**while(VALUE>0)**	

```
20              {
21                VALUE--;
22              }
23          }
```

Circuit Configuration:



2. Program to read the status of toggle switch using ARM LPC 2148.

Line #	Program	Comments
1	`#include<lpc214x.h>`	
2	`int main()`	
3	`{`	
4	`int temp = 0;`	
5	`IO0DIR = 0x000000FF;`	/* Port0 00-07 as output*/
6	`while(1)`	/* Infinite loop */
7	`{`	
8	`temp = IO0PIN ;`	
9	`temp = temp>>16;`	
10	`IOSET0 = temp;`	
11	`IOCLR0 =~temp;`	
12	`}`	
13	`}`	

Result:

Exercise Program

I. Generate an asymmetrical square wave at the lowest four pins of LPC2148.

7. DC MOTOR CONTROL SYSTEM

1. Program to control the direction and speed of the DC motor using ARM LPC 2148.

Line #	Program	Comments
1	**#include<LPC214X.H>**	
2		
3	**#define O1 0X00E00000**	
4	**#define O2 0X00D00000**	
5	**#define O3 0X00B00000**	
6	**#define O4 0X00700000**	
7		
8	**#define I1 0x000E0000**	
9	**#define I2 0x000D0000**	
10	**#define I3 0x000B0000**	
11	**#define I4 0x00070000**	
12		
13	**#define CLR 0x00F00000**	
14		
15	**unsigned int PWM1 = 0x00000001;**	// PWM1
16	**unsigned int PWM2 = 0x00000080;**	// PWM2
17	**unsigned int ENBL = 0x00000002;**	// ENABLE
18		
19	**char scan (int);**	

```
20          void delay(int);
21
22          int main(void)
23          {
24            IO0DIR = 0X00F00083;
25            while(1)
26               {
27                   IO0SET |= ENBL;
28                   IO0CLR = CLR;
29                   IO0SET = O1;
30
31                   while(scan(I1))                    //        S1
32                        {
33                        IO0SET |= PWM1;
34                        IO0CLR |= PWM2;
35                        }
36
37                   while(scan(I2))                    //        S5
38                       {
39                        IO0CLR |= PWM2;
40                        IO0SET |= PWM1;
41                        delay(15);
```

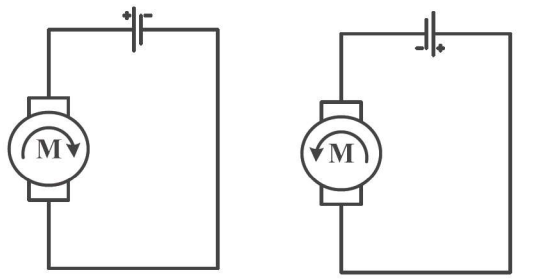
```
42                  IO0CLR |= PWM1;
43                  delay(5);
44                  }
45
46              IO0CLR = CLR;
47              IO0SET = O2;
48
49              while(scan(I1))                 //      S2
50                  {
51                  IO0SET |= PWM2;
52                  IO0CLR |= PWM1;
53                  }
54
55              while(scan(I2))                 //      S6
56                  {
57                  IO0CLR |= PWM2;
58                  IO0SET |= PWM1;
59                  delay(10);
60                  IO0CLR |= PWM1;
61                  delay(10);
62                  }
63
```

```
64                  IO0CLR = CLR;
65                  IO0SET = O3;
66
67                  while(scan(I2))                 //          S7
68                    {
69                     IO0CLR |= PWM2;
70                     IO0SET |= PWM1;
71                     delay(5);
72                     IO0CLR |= PWM1;
73                     delay(15);
74                     }
75              }
76          }
77
78          char scan(int keystatus)                /* scanning a  key*/
79          {
80              while((IO0PIN & 0X000F0000)==keystatus)
81              {
82                      return(1);
83              }
84              return(0);
85          }
```

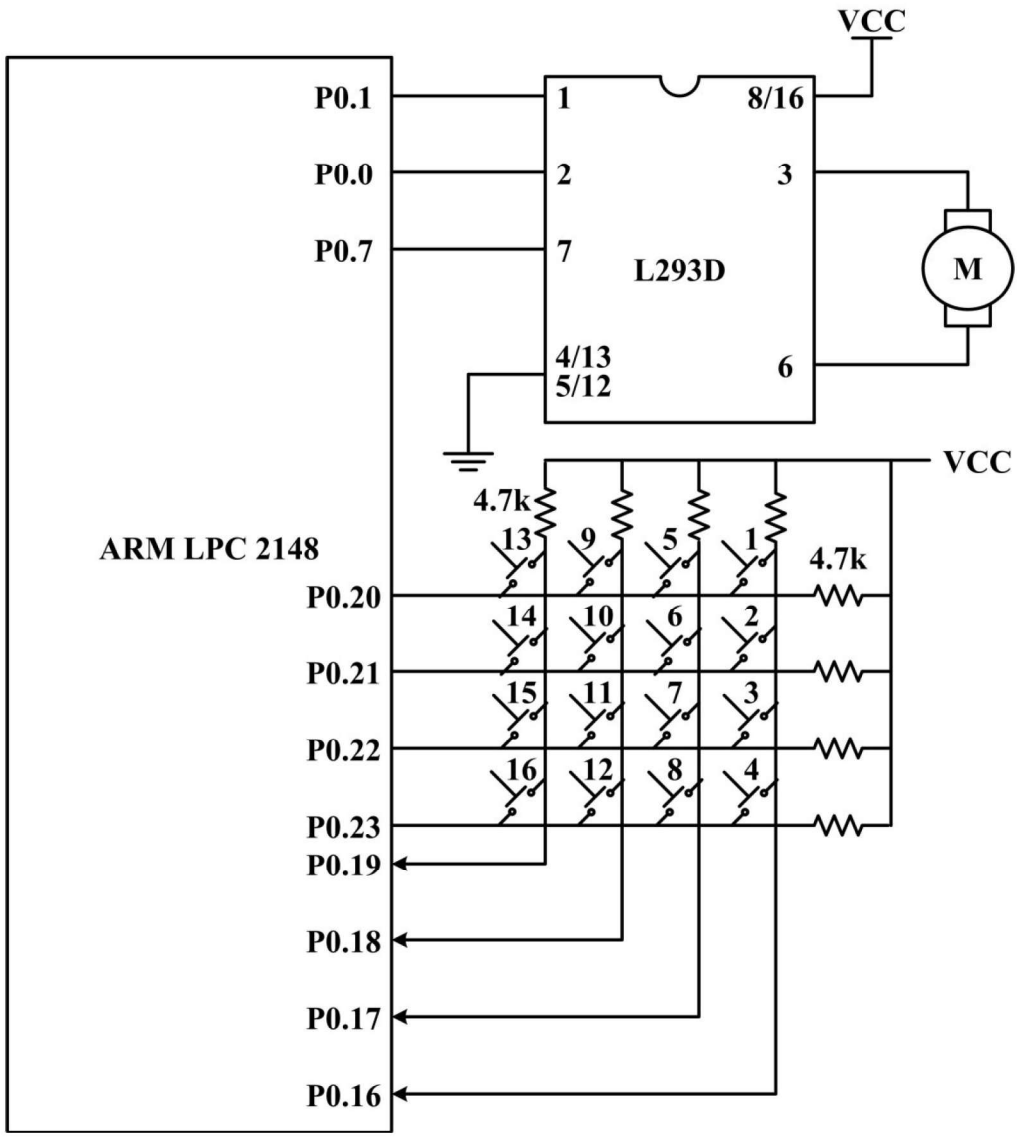
```
86
87      void delay(int n)                          /* generates one
                                                   millisecond delay  */
88      {
89          int i,j;
90          for (i=1; i<=n; i++)
91              for(j=0; j<=10000; j++);
92      }
```

Circuit Configuration:

DC Motor Rotation:



Clockwise rotation **Anticlockwise rotation**


VCC
P0.1
1
8/16
P0.0
2
3
P0.7
7
L293D
M
4/13
5/12
6
VCC
4.7k
13
9
5
1
4.7k
ARM LPC 2148
P0.20
14
10
6
2
P0.21
15
11
7
3
P0.22
16
12
8
4
P0.23
P0.19
P0.18
P0.17
P0.16

2. Program to understand Matrix Keyboard interface to ARM LPC 2148.

Line #	Program	Comments
1	#include<lpc214x.h>	
2		
3	#define O1 0X00E00000	
4	#define O2 0X00D00000	
5	#define O3 0X00B00000	
6	#define O4 0X00700000	
7		
8	#define I1 0x000E0000	
9	#define I2 0x000D0000	
10	#define I3 0x000B0000	
11	#define I4 0x00070000	
12		
13	#define CLR 0x00F00000	
14		
15	char scan (int);	
16	void delay(int);	
17		
18	int main(void)	
19		
20	{	

```
21          IO0DIR = 0X00F0FFFF;
22          while(1)
23              {
24                  IO0CLR = CLR;
25                  IO0SET = O1;
26                  delay(10);
27                  if(scan(I1))IO0SET = 0x00000001;        //  S1
28                  if(scan(I2))IO0SET = 0x00000010;        //  S5
29                  if(scan(I3))IO0SET = 0x00000100;        //  S9
30                  if(scan(I4))IO0SET = 0x00001000;        //  S13
31                  IO0CLR = CLR;
32                  IO0SET = O2;
33                  if(scan(I1))IO0SET = 0x00000002;        //  S2
34                  if(scan(I2))IO0SET = 0x00000020;        //  S6
35                  if(scan(I3))IO0SET = 0x00000200;        //  S10
36                  if(scan(I4))IO0SET = 0x00002000;        //  S14
37                  IO0CLR = CLR;
38                  IO0SET = O3;
39                  if(scan(I1))IO0SET = 0x00000004;        //  S3
40                  if(scan(I2))IO0SET = 0x00000040;        //  S7
41                  if(scan(I3))IO0SET = 0x00000400;        //  S11
42                  if(scan(I4))IO0SET = 0x00004000;        //  S15
```

```
43                  IO0CLR = CLR;
44                  IO0SET = O4;
45                  if(scan(I1))IO0SET = 0x00000008;        //    S4
46                  if(scan(I2))IO0SET = 0x00000080;        //    S8
47                  if(scan(I3))IO0SET = 0x00000800;        //    S12
48                  if(scan(I4))IO0SET = 0x00008000;        //    S16
49              }
50          }
51
52          char scan(int keystatus)                        /* scanning a a key    */
53           {
54                while((IO0PIN & 0X000F0000)==keystatus)
55                {
56                     delay(50);
57                     if((IO0PIN &
                       0X000F0000)==0X000F0000)return(1);
58                 }
59                return(0) ;
60          }
61
62          void delay(int n)                               /* generates one
                                                            millisecond delay  */
```

```
63          {
64                  int i,j;
65                  for (i=1; i<=n; i++)
66                          for(j=0; j<=10000; j++);
67          }
```

Exercise Program

1. **Write a program to rotate the DC motor for variable speed using LPC2148, when switches S10, S11, and S12 are pressed.**

8. STEPPER MOTOR CONTROL SYSTEM

1. Program to control the direction and speed of the stepper motor using ARM LPC 2148.

Line #	Program	Comments
1	**#include<LPC214X.H>**	
2		
3	**#define O1 0X00E00000**	
4	**#define O2 0X00D00000**	
5	**#define O3 0X00B00000**	
6	**#define O4 0X00700000**	
7		
8	**#define I1 0x000E0000**	
9	**#define I2 0x000D0000**	
10	**#define I3 0x000B0000**	
11	**#define I4 0x00070000**	
12		
13	**#define CLR 0x00F00000**	
14	**unsigned char CLOCK[4] = {0x08,0x04,0x02,0x01};**	
15	**unsigned char ANTI_CLOCK[4] = {0x01,0x02,0x04,0x08};**	
16		
17	**char scan (int);**	
18	**void delay(int);**	
19		

```
20      int main(void)
21      {
22          IO0DIR = 0X00F000FF;
23          while(1)
24          {
25              unsigned char I;
26              IO0CLR = CLR;
27              IO0SET = O1;
28              while(scan(I1))                    //    S1
29              {
30                  for(I=0;I<4;I++)
31                  {
32                      IO0SET |= CLOCK[I];
33                      IO0CLR |= ~CLOCK[I];
34                     delay(5);
35                  }
36              }
37              while(scan(I2))                    //    S5
38              {
39                  for(I=0;I<4;I++)
40                  {
41                      IO0SET |= CLOCK[I];
```

```
42                      IO0CLR |= ~CLOCK[I];
43                      delay(10);
44                  }
45              }
46              IO0CLR = CLR;
47              IO0SET = O2;
48              while(scan(I1))                 //  S2
49              {
50                  for(I=0;I<4;I++)
51                  {
52                      IO0SET |= ANTI_CLOCK[I];
53                      IO0CLR |= ~ANTI_CLOCK[I];
54                      delay(5);
55                  }
56              }
57               while(scan(I2))                //  S6
58              {
59                  for(I=0;I<4;I++)
60                  {
61                      IO0SET |= CLOCK[I];
62                      IO0CLR |= ~CLOCK[I];
63                  delay(50);
```

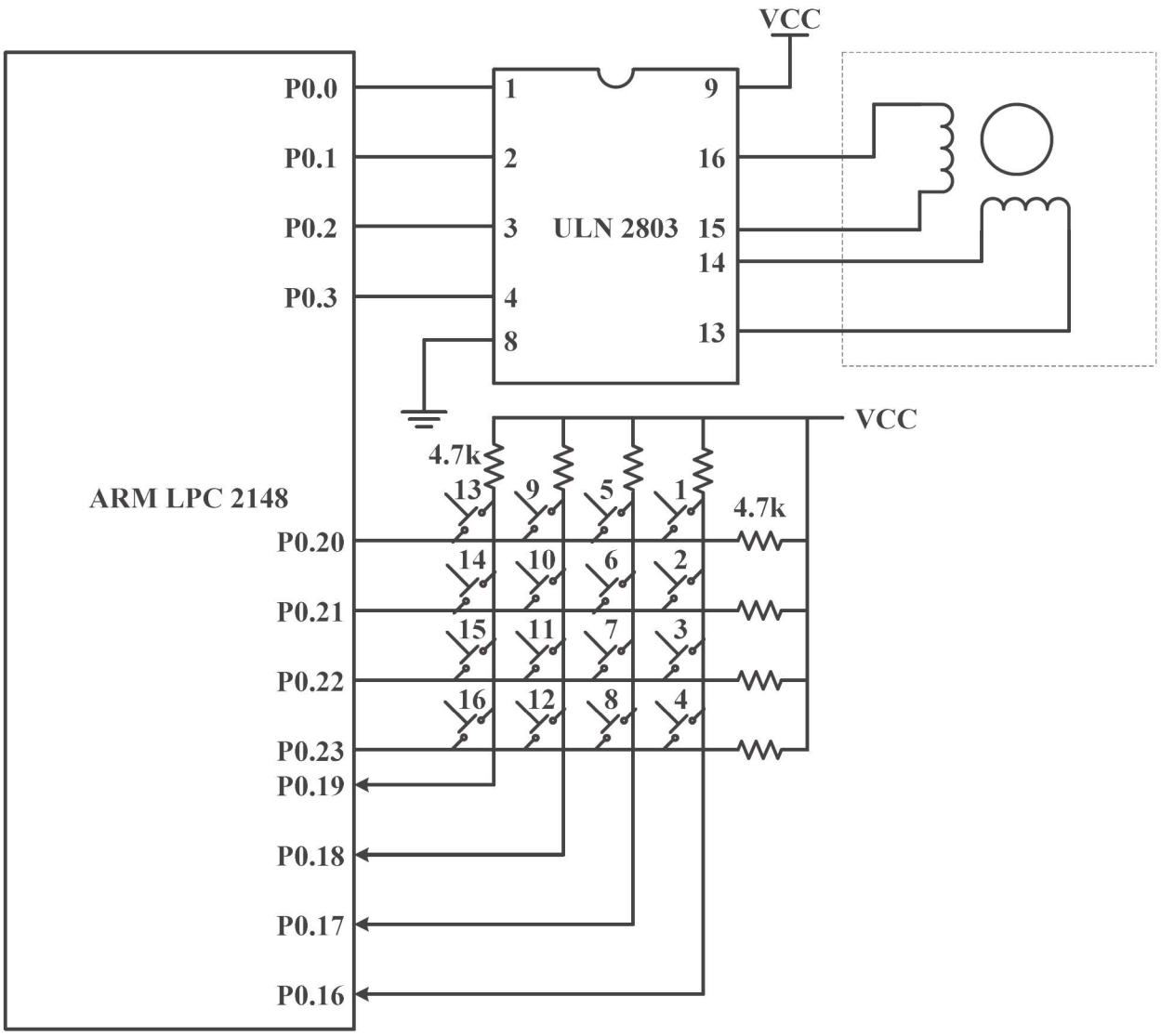
```
64                          }
65                  }
66                  IO0CLR = CLR;
67                  IO0SET = O3;
68                  while(scan(I2))                         //      S7
69                  {
70                          for(I=0;I<4;I++)
71                          {
72                                  IO0SET |= CLOCK[I];
73                                  IO0CLR |= ~CLOCK[I];

74                                  delay(90);
75                          }
76                  }
77          }
78  }
79
80  char scan(int keystatus)                                /* scanning a a key      */
81  {
82          while((IO0PIN & 0X000F0000)==keystatus)
83           {
84                  return(1);
```

```
85              }
86              return(0);
87      }
88
89      void delay(int n)                       /* generates one
                                                   millisecond delay  */
90      {
91              int i,j;
92              for (i=1; i<=n; i++)
93                  for(j=0; j<=10000; j++);
94      }
```

Result:

Circuit Configuration:



4-Step Sequence:

Step #	*Winding A*	*Winding B*	*Winding C*	*Winding D*
1	*1*	*0*	*0*	*0*
2	*0*	*1*	*0*	*0*
3	*0*	*0*	*1*	*0*
4	*0*	*0*	*0*	*1*

Exercise Program

1. **Write a program to rotate the stepper motor clockwise and anticlockwise 90 degree when switches S8 and S9 are pressed respectively.**

2. **Write a program to rotate the stepper motor clockwise 135 degree when switch S12 is pressed .**

9. WAVEFORM GENERATION

1. **Program to generate different waveforms by pressing various keys using ARM LPC 2148.**

Line #	Program	Comments
1	**#include<LPC214X.H>**	
2		
3	**#define O1 0X00E00000**	
4	**#define O2 0X00D00000**	
5	**#define O3 0X00B00000**	
6	**#define O4 0X00700000**	
7		
8	**#define I1 0x000E0000**	
9	**#define I2 0x000D0000**	
10	**#define I3 0x000B0000**	
11	**#define I4 0x00070000**	
12		
13	**#define CLR 0x00F00000**	
14		
15	**char scan (int);**	
16	**void DELAY(int);**	
17		
18	**int main(void)**	
19	**{**	

```
20      unsigned char  table[180] =                    //      Array
        {128,132,136,141,154,150,154,158,163,167,171,175,18
        0,184,188,192,195,199,203,206,210,213,216,220,223,22
        6,228,231,234,236,238,241,243,244,246,247,248,249,25
        0,251,252,253,254,255,255,255,255,255,254,254,253,25
        2,251,249,246,244,243,241,238,236,234,231,228,226,22
        3,220,216,213,210,206,203,199,195,192,188,184,180,17
        5,171,167,163,158,154,150,145,141,136,132,128,123,11
        9,114,110,105,101,97,92,88,84,80,75,71,67,64,60,56,52,
        49,45,42,39,35,32,29,27,24,21,19,17,14,12,11,9,7,6,4,3,
        2,1,1,0,0,0,0,0,0,0,0,1,1,2,3,4,6,7,9,11,12,14,17,19,21,24
        ,27,29,32,35,39,42,45,49,52,56,60,64,67,71,75,80,84,88,
        92,97,101,105,110,114,119,123,128};
21      IO0DIR = 0X00F0FF00;
22      while(1)
23          {
24           IO0CLR = CLR;
25           IO0SET = O1;
26           while(scan(I1))                           //S1      SINE WAVE
27                {
28                 unsigned char I;
29                 unsigned int SINE,A;
30                 for (I=0;I<180;I++)
```

```
31                  {
32                      DELAY (5);
33                      SINE = (table[I]);
34                      for(A=0;A<8;A++)
35                          {
36                              SINE = SINE<<1;
37                          }
38                      IO0SET |= SINE;
39                      IO0CLR |= ~SINE;
40                  }
41              }
42          IO0CLR = CLR;
43          IO0SET = O2;
44          while(scan(I1))                          //S2  TRIANGULAR
                                                     WAVE
45            {
46                  unsigned char J;
47                  unsigned int TRI,B;
48                  for (J=0;J<255;J++)
49                  {
50                      DELAY (2);
51                      TRI = J;
```

```
52                          for(B=0;B<8;B++)
53                          {
54                              TRI = TRI<<1;
55                          }
56                          IO0SET |= TRI;
57                          IO0CLR |= ~TRI;
58                  }
59                  for (J=255;J>0;J--)
60                  {
61                          DELAY (2);
62                          TRI = J;
63                          for(B=0;B<8;B++)
64                          {
65                              TRI = TRI<<1;
66                          }
67                          IO0SET |= TRI;
68                          IO0CLR |= ~TRI;
69                  }
70              }
71          IO0CLR = CLR;
72          IO0SET = O3;
73          while(scan(I1))                         //S3   SQUARE WAVE
```

```
74                  {
75                      IO0SET |= 0x0000FF00;
76                      DELAY (500);
77                      IO0CLR |= 0x0000FF00;
78                      DELAY (500);
79                  }
80              }
81      }
82
83      char scan(int keystatus)                    /* scanning a a key      */
84      {
85          while((IO0PIN & 0X000F0000)==keystatus)
86          {
87              return(1);
88          }
89          return(0);
90      }
91
92      void DELAY(int n)                           /* generates one
                                                    milisecond delay  */
93      {
94      int i,j;
```

```
95          for (i=1; i<=n; i++)
96                  for(j=0; j<=500; j++);
97          }
```

Result:

Calculation:

Angle vs. Voltage magnitude for Sine wave.

$Angle\ \theta$ $(in\ degrees)$	$sin\theta$	$V_{out}(voltage\ magnitude)$ $5V + (5V \times sin\theta)$	$Values\ sent\ to\ DAC\ (decimal)$ $(Voltage\ Mag. \times 25.6)$

Exercise Program

1. **Write a program to generate staircase wave, when switch S11 is pressed.**

2. **Write a program to generate positive ramp, when switch S10 is pressed.**

10. LED DISPLAY INTERFACE

1. Program for LED display using ARM LPC 2148.

Line #	Program	Comments
1	`#include <LPC214x.H>`	
2		
3	`unsigned int srck = 0x00000002;`	
4	`unsigned int rck = 0x00000001;`	
5	`unsigned int ser = 0x00000004;`	
6		
7	`void delay(int);`	
8	`void convert_display(unsigned char);`	
9	`void Clear (void);`	
10	`int display(unsigned char);`	
11	`unsigned char disp[10] = {0x40,0xcf,0xa4,0x30,0x19,0x12,0x02,0xf8,0x00,0x10};`	
12	`unsigned char count = 0;`	
13		
14	`int main()`	
15	`{`	
16	`IO0DIR |= 0x00000007;`	
17	`Clear();`	
18	`while(1)`	

```
19          {
20              convert_display(count);
21              count++;
22              delay(100);
23              Clear();
24          }
25      }
26
27      void Clear (void)
28      {
29          unsigned int x;
30          for (x=0;x<6;x++)
31          {
32              display(0xFF);
33          }
34      }
35
36      void delay(int n)                    /* generates one
                                             millisecond delay */
37      {
38          int i,j;
39          for (i=1; i<=n; i++)
```

```
40                  for(j=0; j<=10000; j++);
41          }
42
43          int display(unsigned char value)
44          {
45                  char m;
46                  int buffer;
47                  buffer = value;
48                  for(m=0;m<8;m++)
49                  {
50                      if(buffer&0x80) IO0SET |= ser;
51                      else IO0CLR |= ser;
52                      IO0SET |= srck;
53                      buffer<<=1;
54                      IO0CLR |= srck;
55                  }
56                  IO0SET |= rck;
57                  ;
58                  IO0CLR |= rck;
59                  return 0;
60          }
61
```

```
62      void convert_display(unsigned char value)
63      {
64          unsigned char x, d1, d2, d3;
65          x = value / 10;                 //  divide by 10
66          d1 = value % 10;                //  save low digit
                                            (reminder of division)
67          d2 = x % 10;
68          d3 = x / 10;                    //  divide by 10 once
                                            more
69          display(disp[d3]);
70          display(disp[d2]);
71          display(disp[d1]);
72          return;
73      }
```

Result:

Exercise Program

I. Write a program to display pressed key on LED display.

Viva Questions

1. Answer the following:
   i) One nibble = _____bits
   ii) One byte = _____bits
   iii) How many K is 1 giga?
2. Convert decimal numbers to hex numbers.
3. Perform addition of hex numbers.
4. Give the example for microcontrollers and microprocessors.
5. The 8051 is an _____-bit microcontroller.
6. List the features of 8051.
7. Name any three 16-bit microcontrollers.
8. What are PSW and SFRs?
9. What is simulator/debugger?
10. What are 8051 data types and directives?
11. What is the size of SP register?
12. SP is incremented by 1 for each _____operation.
13. On power–up , The 8051 uses_____bank.
14. List the 16 –bit registers in the 8051.
15. What are the four fields of assembly language instructions?
16. MOV A, #40 h requires _____bytes.
17. What is machine cycle for 8051?
18. What is delay? Why do we need it?
19. In what ways instructions LJMP and SJMP are different?
20. Why do we need subroutines?
21. What is the difference between RET and RETI instructions?
22. How many ports are available in 8051?
23. Which 8051 ports need pull-up resistors to function as an I/O ports?
24. Write simple instruction to send 99h to ports P1 and P2.
25. Which pins of port 3 cater to interrupts?
26. Port 2 is also designated as _____ of 8051.
27. List the addressing modes provided by 8051 with examples.
28. What is done by the following instructions?

```
MOV A, P1
JB  ACC.1, HERE
```

29. Is following instruction valid?

```
MOV C, A
```

30. What is the difference between the operations of a timer and a counter?
31. What is the use of TMOD register?
32. What is the difference between mode 1 and mode 2 operations of timer?
33. What is the address of TCON register?
34. Distinguish between half duplex and full duplex mode of communication with an example.
35. What is the function of MAX 232 chip?
36. State the most important signals of RS232.
37. What is baud rate?
38. What is the roll of SBUF in serial communication?
39. Show the interrupt vector table for the 8051.
40. What is ARM7TDMI.
41. List the features of ARM microcontroller.
42. ARM has _____ registers each of which is_____bit long.
43. What is the difference between little endian and big endian format.
44. What is done by the following instruction of the ARM?

    MVN R0, R9
45. What are the types of the stacks available in ARM?
46. List any three features of LPC2148.
47. What is the use of PWM unit?
48. What is the difference between CCLK and PCLK?
49. How to set up the communication between PC and microcontroller?
50. How does the prescaler in time unit function?

References

[1]. Mazidi M. A. & J. G. Mazidi, *The 8051 Microcontroller and embedded systems*, Pearson. 2002

[2]. LPC21XX User Manual. 2007

[3]. Ramani Kalpathi Microcontrollers & Applications, Sanguine Technical Publishers, 2008