# BHUBANANANDA ODISHA SCHOOL OF ENGINEERING CUTTACK

# DEPARTMENT OF ELECTRONICS AND TELECOMMUNICATION ENGINEERING

# LAB MANUAL

**Year & Semester: 3RD Year, VI Semester**

**Name: MICROPROCESSOR & MICROCONTROLLER LAB**

**LIST OF EXPERIMENTS**

| SL.NO. | NAME OF THE EXPERIMENT |
|---|---|
| 1. | Study the architecture diagram of 8051 micro-controller. |
| 2.A | Write a program for addition of two 8- bit number whose sum is 8-bit. |
| 2.B | Write a program for subtraction of two 8- bit number. |
| 2.C | To write a program for multiplication of two 8- bit numbers. |
| 2.D | To write a program for division of two 8- bit numbers. |
| 3.A | Write a program for arranging series of number in ascending order. |
| 3.B | Write a program for arranging series of number in descending order |
| 4. | Write a program for 8 bit digital output-LED interface. |
| 5.A | Write a assembly program for relay interface. |
| 5.B | Write a simple program to interface buzzer sound. |
| 6. | To write an assembly language program for character based LCD Interface. |
| 7. | To interface stepper motor with 8051 parallel port and to vary speed of motor and direction of motor. |
| 8. | Write a program to generate delay subroutine. |
| 9. | Write simple program to interface traffic light systems. |
| 10. | Write a simple program for blinking of two separate LED. |

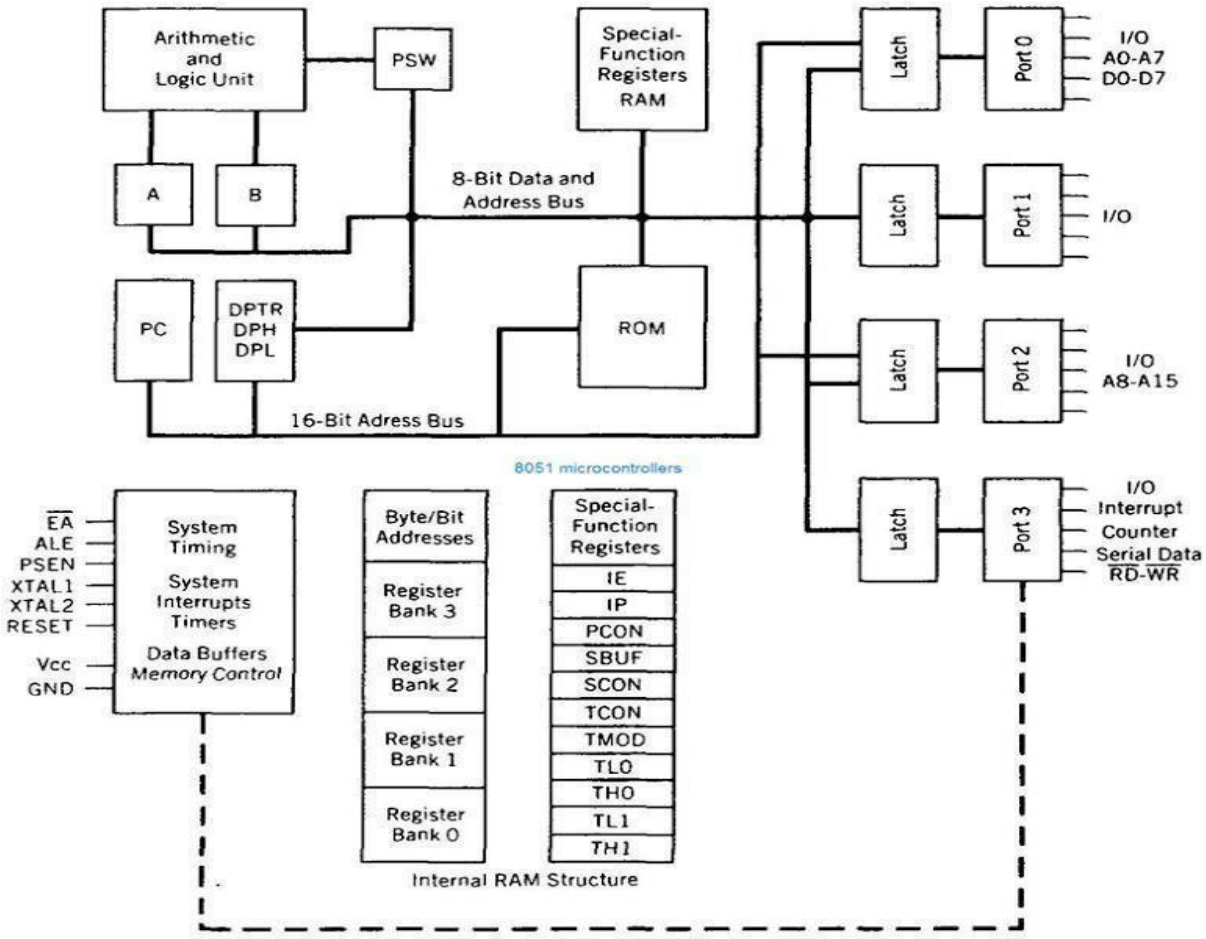
# EXPERIMENT NO. – 1

**AIM OF THE EXPERIMENT:**

Study the architecture diagram of 8051 micro-controller.

**THEORY:**

**8051 ARCHITECHTURE:**




**ALU:** Arithmetic & logical unit is a main part of microcontroller which is used for the arithmetic operation such as addition, subtraction, multiplication, division & logical operations (such as AND, OR, NAND, NOR, NOT, increment, decrement).

**A-ACCUMULATOR:** This register is used for arithmetic operation in holding of one of the operand in the operation. It is 8bit & also bit addressable registers. "A" represents as 8 bit accumulator which is used for holding operands in arithmetic operation.

**B-REGISTER:** It is an 8bit addressable two register which is used only for operate two

**PC PROG:** It is a 16 bit special purpose register which is used to hold the address of next instruction to be executed from ROM. So it can access program address from 0000 H & FFFFH , it. has 68 KB .

**RAM:** It is a non-volatile read only memory which can written data even if when power OFF. In this the user can only read the data.

**ROM:** Here 4KB, 8KB, 16KB, 32KB, 64 KB chip ROM is available.
Maximum ROM space is 64KB because 16 BIT address line is available in 8051.
Starting address for ROM is 0000H.

**8051 FLAGBITS & PSW REGISTERS:** It is used to indicate arithmetic condition of ACC. Flag register in 8051 is called as program status word (PSW). This special function register PSW is also bit addressable & 8 bit wide means each bit can be SET or RESET independently. There are 4 flags in 8051.

**PARITY FLAG:** 1- ODD NUMBER OF 1 IN ACC
2- EVEN NUMBER IF 1 IN ACC
It will active to when the result contains even no. If 1 & active to 1, when the result contains odd no. of 1.

**OVERFLOW FLAG:** This is used to detect in over in signed arithmetic operation. This is similar to carry flag but difference is only that carry flag is used for unsigned operation.

| **RS1** | **RS0** | **REGISTER BANK SELECT** |
|---|---|---|
| 0 | 0 | BANK 0 (00H-07H) LET HERE FOR BANK 1 |
| 0 | 1 | BANK 1 (08H-0CH) SET B PSW0 = 3 (MEANS S1 S0 =1 ) |
| 1 | 0 | BANK 2 (10H-17H) CLR PSW00.4 (CONEOUS RS1=0 ) |
| 1 | 1 | BANK 3 (18H-18H) |

**AUXILARY CARRY:** Auxiliary carry active when carry is taken from D3 BIT TO D4.

**CARRY FLAG:** It wills active when the result generates carry.

**RAM:** RAM is used for both READ &WRITE operation. It is volatile in nature. It is a special function register which has 128 BYTE storage capacity. It contains & register bank each register have 8 BIT registers R0 TO R7.

**BANK 0:** It contains 8 bit register i.e., R0 TO R7.

| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

RAM contains special function register i.e.

TH1- TIMER 1 HIGHER ORDER BYTES – 80H

TL1 - TIMER 1 LOW ORDER BYTES -86H

SBUF- SERIAL DATA BUFFER- 99H

PCON- POWER CONTROL – 87H

**STACK POINTER:** It is a special purpose 8 bit register which is used in 8051 for hold on internal ram address i.e. hold at stack top .Here stack is used to store data temporarily based ON LIFO principle.

**DPTR:** Its Q made up of two 8BIT register name DPH & DPL which are used furnish memory address. For internal & external code & external data address. So it can be specified by 16 BIT name or by each individual.

**TMOD**

**COUNTER:** It consist of two 16 BIT counter to (TL0,TH0) & T1(TL1,TH1) provided for general use of the programmer external. It is used to count the intervals clock pulses.

**TIMER:** Timer is used to provide time signal such as TCON & TMOD signal. TMOD IS 4BIT controlling signal TCON is 8 BIT controlling signal.

**I/P, O/P PORT:** There are 4 I/P, O/P port available i.e. P0, P1, P2, P3. Each port is 8 BIT wide & has special function register which are bit addressable means each BIT can SET OR RESET by the BIT instruction.

**PORT 0:** Performed dual work. It consist of lower order 8 BIT address bus which time multiplex with 8 BIT data BIT i.e., P.02 to P.07.
**PORT 1:** It is used for only I/P, O/P connections. It is also 8 BIT wide.

**PORT 2:** It can use as I/O port also used for highest order address bus.

**PORT 3:** It also has dual function 8 BIT wide register. Its pin has specific function i.e.

P3.0- RXD (SERIAL I/P FOR ASYNCHRONOUS COMMUNICATION)
(SERIAL O/P FOR SYNCHRONOUS COMMUNICATION)

P3.1-TXD – SERIAL DATA TRANSMIT

P3.2-INT0 – EXTERNAL INTERRUPT 0.

P3.3-INT1 – EXTERNAL INTERRUPT 1.

P3.4-T0 – CLK INPUT FOR COUNTER 0.

P3.5-T1 – CLK OUTPUT FOR COUNTER 1.

P3.6 – WRITE OPERATION - WR

P3.7 – READ OPERATION

**CONCLUSION:**

# EXPERIMENT NO.:2(A)

AIM OF THE EXPERIMENT

Write a program for addition of two 8- bit number whose sum is 8-bit.

**EQUIPMENT REQUIRED:**
8051 Microcontroller trainer kit

**THEORY:**
ADD 49H & 56H
The first number 49H each in the external memory location 9001H & the second no. 56H each in the external memory location 9002H. The result is to be stored in the external memory.

**PROCEDURE:**

| Memory address | Machine code | Mnemonics | Operands | Comments |
|---|---|---|---|---|
| 8000 | 90,90,01 | MOV | DPTR, #9001 | Load 16-bit constant 9001 DPTR |
| 8003 | E0 | MOV X | A@ DPTR | Move the content of external memory 9001 into accumulator. |
| 8004 | F5, 0B | MOV | B,A | Move accumulator to B |

**PROGRAM:**

| | | | | |
|---|---|---|---|---|
| 8006 | A3 | INC | DPTR | Increment DPTR |
| 8007 | E0 | MOV X | A, @DPTR | Move 2nd data into accumulator |
| 8008 | 25,0B | ADD | A,B | ADD 'B' register with accumulator |
| 800A | A3 | INC | DPTR | Increment DPTR |
| 800B | F0 | MOV X | @DPTR,A | Store result into 9003H |

**INPUT: 9001H=49H**
**9002H=56H**

**OUTPUT:**
**9003H=9FH**

**CONCLUSION:**

# EXPERIMENT NO.:2(B)

**AIM OF THE EXPERIMENT**

Write a program for subtraction of two 8- bit number.

**EQUIPMENT REQUIRED:**
8051 Microcontroller trainer kit

**THEORY:**

EFH and 45H the first number. EFH is in the accumulator and the second number. 45H is in the internal RAM R0 register and the result is stored in the internal RAM R1 register.

**PROCEDURE:**

**STEP 1- STORAGE OF PROGRAM MACHINE CODES:**
Machine codes of the program to be executed, should be stored in their memory available on 8051 trainer kit as
**RESET→ EXMEM→** starting address of program (2000)→**NEXT→** Now enter all themachine codes, one after the other followed by the key **'NEXT'.**

**STEP 2-EXECUTION OF PROGRAM:**
The program can be executed as
**FILL→ GO→** Starting address of the program (2000)→**FILL**

**STEP 3-VERIFICATION OF OUTPUT:**
Output of the executed program can be verified as
**RESET→ EXMEM→**type the memory address where result will be stored →**NEXT**

**PROGRAM:**

| Memory address | Machine code | Mnemonics | Operands | Comments |
|---|---|---|---|---|
| 8000 | 74, EF | MOV | A, #EFH | Load EFH into accumulator |
| 8002 | 78, 45 | MOV | R0 #45H | Load 45H in R0 register |

| | | | | |
|---|---|---|---|---|
| 8004 | 78 | SUB | A, R0 | Subtract the content of R0 from accumulator |
| 8005 | F9 | MOV | R1, A | Move the content of accumulator to R1 register |
| 8006 | 02, 00, 00 | LJMP | 00 | |

**INPUT: A= EFH** **OUTPUT: AA**

**R0= B= 45H**

**CONCLUSION:**

# EXPERIMENT NO.: 2(C)

**AIM OF THE EXPERIMENT:**

To write a program for multiplication of two 8- bit numbers.

**EQUIPMENT REQUIRED:**
8051 Microcontroller trainer kit

**THEORY:**

Multiply 24H and 12H the first number 24H is in the accumulator & second number 12H is in the register B. Result is stored in R0 &R1 internal RAM location. The LSB of the address is in the R0 and MSB is in the R1 memory location.

**PROCEDURE:**

| Memory address | Machine code | Mnemonics | Operands | Comments |
|---|---|---|---|---|
| 8000 | 74, 24 | MOV | A, #24H | Load accumulator which 1st number |
| 8002 | 75, 0B, 12 | MOV | B, # 12 | Load register which is 2nd number of accumulator. |
| 8005 | AU | MUL | A, B | Multiply accumulator or with register and the result is stored accumulator. |

**STEP 1- STORAGE OF PROGRAM MACHINE CODES:**
Machine codes of the program to be executed, should be stored in their memory available on 8051 trainer kit as

**RESET→ EXMEM→** starting address of program (2000)**→NEXT→** Now enter all themachine codes, one after the other followed by the key **'NEXT'.**

**STEP 2-EXECUTION OF PROGRAM:**
The program can be executed as
**FILL→ GO→** Starting address of the program (2000)**→FILL**

**STEP 3-VERIFICATION OF OUTPUT:**

Output of the executed program can be verified as

**RESET→ EXMEM→** type the memory address where result will be stored **→NEXT**

**PROGRAM:**

| | | | | |
|---|---|---|---|---|
| 8006 | 78 | MOV | R0, A | Store LSB in R0 register. |
| 8007 | A9,0B | MOV | R1, B | Store MSB in R1 register. |
| 8009 | 02,00,00 | JMP | 0000 | ------- |

**INPUT:** **A= 24H** **OUTPUT: 0288**

**B= 12H**

**R0 = 02**

**R1 = 88**

**CONCLUSION:**

# EXPERIMENT NO:-2(D)

**AIM OF THE EXPERIMENT:**

To write a program for division of two 8- bit numbers.

**EQUIPMENT REQUIRED:**
8051 Microcontroller trainer kit

**THEORY:**

Divide 16H and 12H is stored in accumulator and the 2nd number 12H in the register B. The result is stored in R0 register and reminder is stored in the R1 register.

**PROCEDURE:**

**STEP 1- STORAGE OF PROGRAM MACHINE CODES:**
Machine codes of the program to be executed, should be stored in their memory available on 8051 trainer kit as
**RESET→ EXMEM→** starting address of program (2000)→**NEXT→** Now enter all themachine codes, one after the other followed by the key **'NEXT'.**

**STEP 2-EXECUTION OF PROGRAM:**
The program can be executed as
**FILL→ GO→** Starting address of the program (2000)→**FILL**

**STEP 3-VERIFICATION OF OUTPUT:**
Output of the executed program can be verified as
**RESET→ EXMEM→** type the memory address where result will be stored →**NEXT**

**PROGRAM:**

| Memory address | Machine code | Mnemonics | Operands | Comments |
|---|---|---|---|---|
| 8000 | 74, 60 | MOV | A, #60H | Load accumulator with 1st number |

| | | | | |
|---|---|---|---|---|
| 8002 | 75, 0B, 12 | MOV | B, # 12H | Load accumulator with is 2nd number. |
| 8005 | 84 | DIV | A, B | Divide accumulator with register and the result is stored in accumulator. |
| 8006 | F8 | MOV | R0, A | Result is in the R0 register. |
| 8007 | F9,08 | MOV | R1, B | Stored reminder in R1. |
| 8009 | 02,00,00 | LJMP | 0000 | ------- |

**INPUT: A= 60H** **OUTPUT: 05**
**B= 12H**

**CONCLUSION:**

# EXPERIMENT NO:-3(A)

**AIM OF THE EXPERIMENT:**

Write a program for arranging series of number in ascending order.

**EQUIPMENT REQUIRED:**
8051 Microcontroller trainer kit

**THEORY:**

Numbers of data byte is on which is stored in the R0 register. The memory location starts from 9000H.

**PROGRAM:**

| Memory address | Machine code | | Mnemonics | Operands | Comments |
|---|---|---|---|---|---|
| 8000 | 78, 08 | - | MOV | R0,#0A | Data 0A is stored in R0 register |
| 8002 | 81 | - | DEC | R0 | Decrement R0 |
| 8003 | 90,00,00 | LOOP1 | MOV | DPTR #9000 | Load 9000 in DPTR |
| 8006 | F8 | - | MOV | A, R0 | Content of R0 is moved to A |
| 8007 | F9 | - | MOV | R1, A | Content of A to R1 register |
| 8008 | E0 | LOOP2 | MOVX | A,@DPTR | Move the content of location which is specified by DPTR to A |
| 8009 | FA | - | MOV | R2,A | Content of A to R2 |
| 800A | A3 | - | INC | DPTR | Increment DPTR |
| 800B | E0 | - | MOVX | A,@DPTR | Move the content of location which is specified by DPTR to A |

| | | | | | |
|---|---|---|---|---|---|
| 800C | 9A | | SUB,B | A,A2 | Compare A with R2 |
| 800D | 50, 08 | - | INC | 8017 LOOP3 | Jump to 8017 if carry equals to A |
| 800F | E0 | - | MOVX | A2,@DPTR | Move the content of location which is |
| | | | | | specified by DPTR to A |
| 8010 | CA | - | XCH | A, R2 | Exchange data in A & R2 if carry is 0 |
| 8011 | F0 | - | | @DPTR, A | Replace carry memory data by decrement data by 1 |
| 8014 | EA | - | DEC | 82 | Decrement data byte 1 |
| 8015 | F0 | - | MOV | A,82 | Move R2 to A |
| 8016 | A3 | - | INC | DPTR | Decrement DPTR |
| 8017 | 09,EE | LOOP3 | DINZ | R,8008, LOOP2 | Decrement R1, if not 0 jump to LOOP3 |
| 8019 | 08,E8 | - | DJNZ | R,8003, LOOP | Decrement R1, if not jump to LOOP3 |
| 801B | 02,00,00 | - | JMP | 0000 | |

**CONCLUSION:**

# EXPERIMENT NO:-3(B)

**AIM OF THE EXPERIMENT:**

Write a program for arranging series of number in descending order.

**EQUIPMENT REQUIRED:**
8051 Microcontroller trainer kit

**THEORY:**

Numbers of data byte is on which is stored in the R0 register, internal RAM memory location starts from 9001H.

**PROCEDURE:**

**STEP 1- STORAGE OF PROGRAM MACHINE CODES:**
Machine codes of the program to be executed, should be stored in their memory available on 8051 trainer kit as
**RESET→ EXMEM→** starting address of program (2000)**→NEXT→** Now enter all themachine codes, one after the other followed by the key **'NEXT'.**

**STEP 2-EXECUTION OF PROGRAM:**
The program can be executed as
**FILL→ GO→** Starting address of the program (2000)**→FILL**

**STEP 3-VERIFICATION OF OUTPUT:**
Output of the executed program can be verified as
**RESET→ EXMEM→** type the memory address where result will be stored **→NEXT**

**PROGRAM:**

| Memory address | Machine code | | Mnemonics | Operands | Comments |
|---|---|---|---|---|---|
| 8000 | 78, 08 | - | MOV | R0,#0A | Data 0A is stored in R0 register |
| 8002 | 18 | - | DEC | R0 | Decrement R1 |
| 8003 | 90,90,00 | LOOP 1 | MOV | DPTR #9000 | Load 9000 in DPTR |
| 8006 | F8 | - | MOV | A, R0 | Content of R0 is moved to A |

| | | | | | |
|---|---|---|---|---|---|
| 8007 | F9 | - | MOV | R1, A | Content of A to R1 register |
| 8008 | F0 | LOOP 2 | MOV X | A,@DPTR | Move the content of location which is specified by DPTR to A |
| 8009 | EA | - | MOV | R2,A | Content of A to R2 |
| 800A | A3 | - | INC | DPTR | Increment DPTR |
| 800B | E0 | - | MOV X | A,@DPTR | Move the content of location which is specified by DPTR to A |
| 800C | 9A | | SUB,B | A,R2 | Subtract A with R2 |
| 800D | 50, 08 | - | INC | 8017 LOOP3 | Jump to 8017 if carry equals to 1 |
| 800F | E0 | - | MOV X | A,@DPTR | Move the content of memory location which is specified by DPTR to A |
| 8010 | CA | - | XCH | A, R2 | Exchange data in A & R2 if carry is 1 |
| 8011 | F0 | - | MOV X | @DPTR, A | Replace carry memory data in A & R2 If carry=1 |
| 8012 | 15,82 | - | DEC | 82 | Decrement data byte 1 |
| 8014 | EA | - | MOV | A, R2 | Move R2 to A |
| 8015 | F0 | - | MOV X | @DPTR,A | Replace memory data by R2 |
| 8016 | A3 | - | INC | DPTR | Increment DPTR |
| 8017 | 09,EF | LOOP3 | DJNZ | R,8003 | Decrement R1, if not 0 jump to LOOP1 |
| 8019 | 08,E8 | - | DJNZ | R,8003, LOOP | Decrement R1, if not jump to LOOP1 |
| 801B | 02,00,00 | - | LJMP | 0000 | |

**CONCLUSION:**

# EXPERIMENT NO:-4

**AIM OF THE EXPERIMENT:**

Write a program for 8 bit digital output-LED interface.

**APPARATUS REQUIRED:**
8051 Trainer Kit and Interfacing Kit.

**THEORY:**
This program reads the status of switches and display on 8 outputs & seven segment display of the kit.

**PROCEDURE:**

**STEP 1- STORAGE OF PROGRAM MACHINE CODES:**
Machine codes of the program to be executed, should be stored in their memory available on 8051 trainer kit as
**RESET→ EXMEM→** starting address of program (2000)→**NEXT→** Now enter all themachine codes, one after the other followed by the key **'NEXT'.**

**STEP 2-EXECUTION OF PROGRAM:**
The program can be executed as
**FILL→ GO→** Starting address of the program (2000)→**FILL**

**STEP 3-VERIFICATION OF OUTPUT:**
Output of the executed program can be verified as
**RESET→ EXMEM→** type the memory address where result will be stored →**NEXT**

**PROGRAM:**

| Memory address | Machine code | | Mnemonics | Operands | Comments |
|---|---|---|---|---|---|
| 3000 | 90 FF 03 | | MOV | DPTR,#0FF03H | ;Initialize CWR of 8255 |
| 3003 | 74 82 | | MOV | A,#82H | ;Port B as input Port |
| 3005 | F0 | | MOV | @DPTR,A | |
| 3006 | 90 FF 01 | LOOP | MOV | DPTR,@0FF01H | |

| | | | | | |
|---|---|---|---|---|---|
| 3009 | E0 | | MOVX | A,@DPTR | ;read status in port B |
| 300A | 54 FF | | ANL | A,@0FFH | |
| 300C | F4 | | CPL | A | |
| 300D | 90 FF 00 | | MOV | DPTR,#0FF00H | ;OUT DATA AT PORT A |
| 3010 | F0 | | MOVX | @DPTR,A | |
| 3011 | FB | | MOV | R3,A | |
| 3012 | 1230 17 | | LCALL | DISP | ;call display routine |
| 3015 | 80 EF | | SJMP | LOOP | |
| 3017 | 90 32 08 | DISP | MOV | DPTR,#3208H | ;store the code value in ram location 3D00h |
| 301A | EB | | MOV | A,R3 | |
| 301B | 54 F0 | | ANL | A,#0F0H | |
| 301D | C4 | | SWAP | A | |
| 301E | 12 0B 77 | | LCALL | 0B77H | |
| 3021 | F0 | | MOVX | @DPTR,A | |
| 3022 | A3 | | INC | DPTR | |
| 3023 | EB | | MOV | A,R3 | |
| 3024 | 54 0F | | ANL | A,#0FH | |
| 3026 | 12 0B 77 | | LCALL | 0B77H | |
| 3029 | F0 | | MOVX | @DPTR,A | |
| 302A | 90 32 00 | | MOV | DPTR,#3200H | |
| 302D | 12 0A 3C | | LCALL | 0A3CH | ;call display routine |
| 3030 | 22 | | RET | | |
| 3200 | 4F 55 54 50 55 | | DFB | 4FH, 55H, 54H, 50, 55H | |
| 3205 | 54 53 20 20 20 | | DFB | 54H, 53H, 20H, 20H, 20H | |
| 320A | 20 20 20 20 20 | | DFB | 20H, 20H, 20H, 20H, 20H | |
| 320F | 20 20 20 20 20 | | DFB | 20H, 20H, 20H, 20H, 20H, | |
| 3214 | 20 20 20 20 20 | | DFB | 20H, 20H, 20H, 20H, 20H | |
| 3219 | 20 20 20 20 20 | | DFB | 20H, 20H, 20H, 20H, 20H | |

**CONCLUSION**

# EXPERIMENT NO:-5(A)

**AIM OF THE EXPERIMENT:**

Write a assembly program for relay interface.

**APPARATUS REQUIRED:**
8051 trainer kit and interfacing kit.

**THEORY:**
This program observes the status of RELAYS on the module. The turning "ON" and " OFF" of the relays can be heard. With the help of multi meter one can check the statusof relay by testing the status of NO & NC of corresponding relay.

**PROCEDURE:**

**STEP 1- STORAGE OF PROGRAM MACHINE CODES:**
Machine codes of the program to be executed, should be stored in their memory available on 8051 trainer kit as
**RESET→ EXMEM→** starting address of program (2000)**→NEXT→** Now enter all themachine codes, one after the other followed by the key **'NEXT'.**

**STEP 2-EXECUTION OF PROGRAM:**
The program can be executed as
**FILL→GO→** Starting address of the program (2000)**→FILL**

**STEP 3-VERIFICATION OF OUTPUT:**
Output of the executed program can be verified as
**RESET→ EXMEM→** type the memory address where result will be stored **→NEXTPROGRAM:**

| Memory address | Machine code | | Mnemonics | Operands | Comments |
|---|---|---|---|---|---|
| 3000 | 90 FF 03 | | MOV | DPTR,#OFF03H | |
| 3003 | 74 80 | | MOV | A,#80H | ; Initialise CWR 8255-1 |
| 3005 | F0 | | MOVX | @DOTR<A | ;Port A as output |
| 3006 | 90 FF 01 | LOOP | MOV | DPTR#OFF01H | |

| | | | | | |
|---|---|---|---|---|---|
| 3009 | 74 55 | | MOV | A,#55H | ;make relay 1 & relay3 on,Relay2 & Relay 4 off |
| 300B | F0 | | MOVX | @DPTR,A | |
| 300C | 12 30 17 | | LCALL | DELAY | |
| 300F | 74 AA | | MOV | A,#0AAH | ;Make relay2&relay4 on Relay1& relay3 off |

**DELAY SUBROUTINE:**

| Memory address | Machine code | | Mnemonics | Operands | Comments |
|---|---|---|---|---|---|
| 3011 | F0 | | MOVX | @DPTR,A | |
| 3012 | 12 30 17 | | LCALL | DELAY | |
| 3015 | 80 EF | | SJMP | LOOP | |
| 3017 | 79 00 | DELAY | MOV | R1,#0H | ;delay |
| 3019 | 7A 00 | DL2: | MOV | R2,#0H | |
| 301B | DA FE | DL1: | DJNZ | R2,DL1 | |
| 301D | D9 FA | | DJNZ | R1,DL2 | |
| 301F | 22 | | RET | | |

**CONCLUSION:**

# EXPERIMENT NO:-5(B)

**AIM OF THE EXPERIMENT-**

Write a simple program to interface buzzer sound.

**COMPONENTS REQUIRED-**

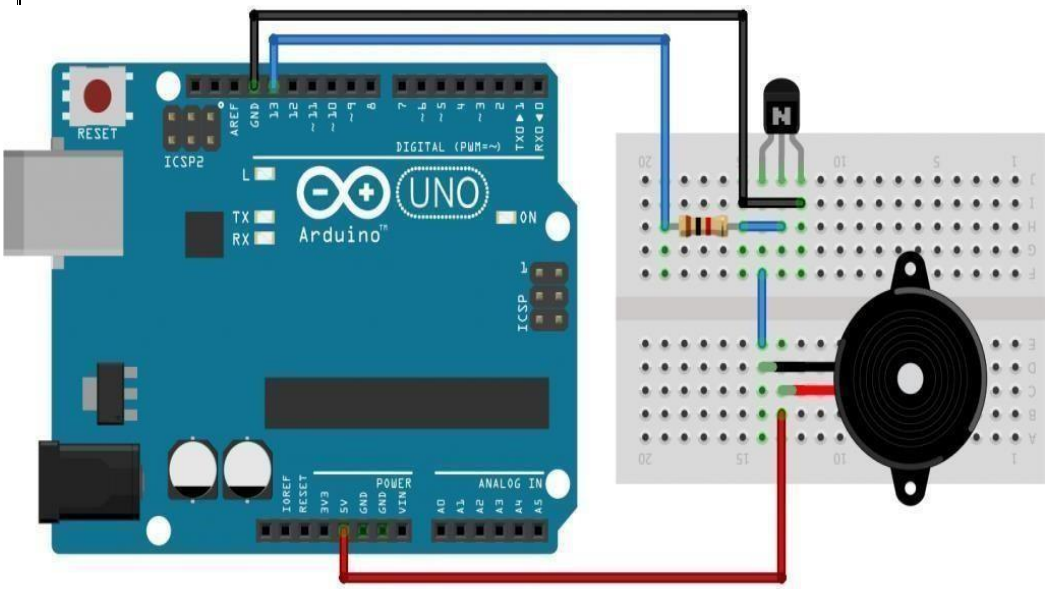
- Arduino UNO
- Piezo Buzzer
- NPN Transistor BC548
- Resistor 1K ohm
- breadboard

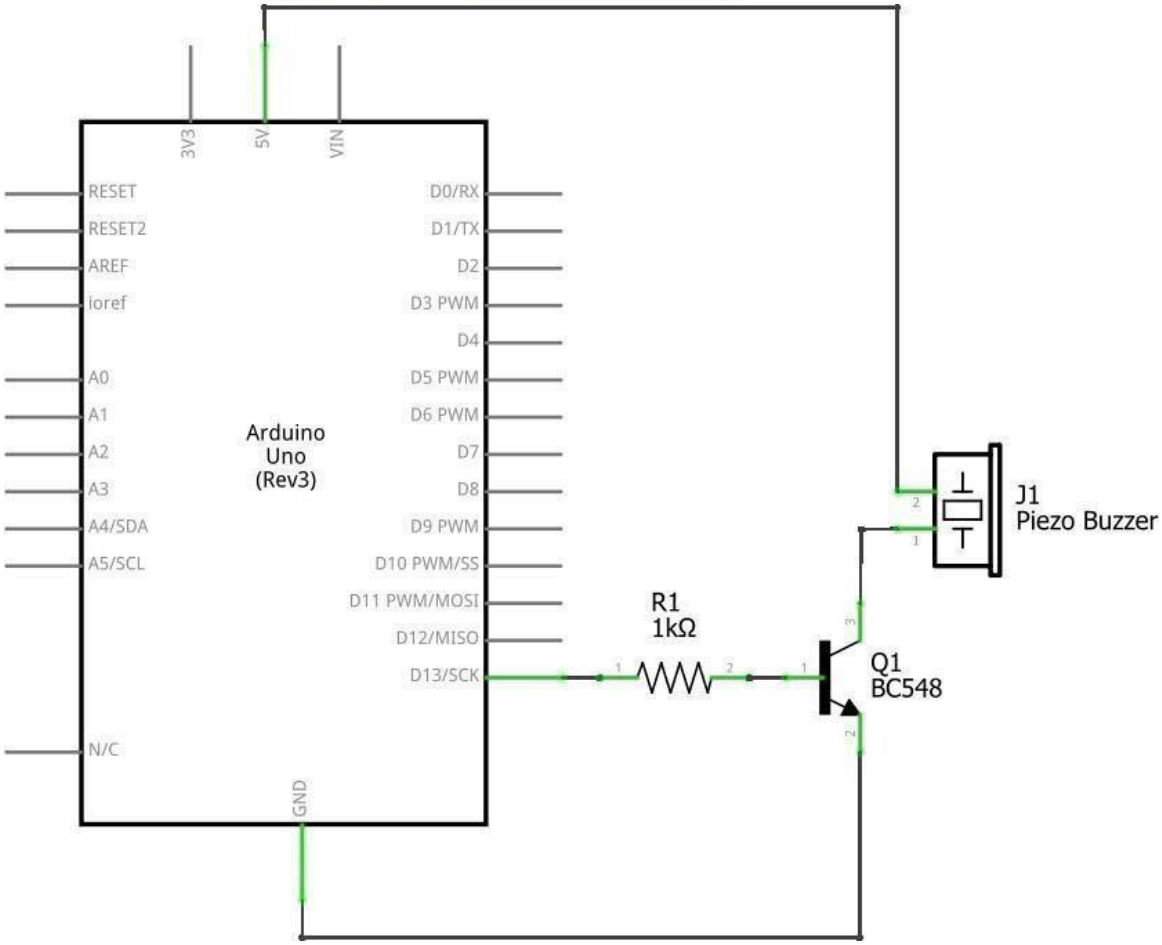
**THEORY:**

A piezo buzzer is generally used to signal user in the form of tone or beep. This type of buzzer widely used in alarm, domestic gadgets or in embedded systems product to provide some kind of indication or alert.. We will design a small transistorized circuit and by providing high signal we can get a tone signal from the piezo buzzer.

- Connect the buzzer to the VCC and collector of the transistor BC548.
- Connect a 1k ohm resistor to the base of the transistor.
- Here transistor works as a switch and by applying high signal to its base triggers the buzzer to beep.

**CIRCUIT:**


RESET
ICSP2
DIGITAL (PWM=~)
UNO
Arduino
TX
RX
ON
ICSP
POWER
ANALOG IN


3V3
5V
VIN
RESET
RESET2
AREF
ioref
A0
A1
A2
A3
A4/SDA
A5/SCL
N/C
D0/RX
D1/TX
D2
D3 PWM
D4
D5 PWM
D6 PWM
D7
D8
D9 PWM
D10 PWM/SS
D11 PWM/MOSI
D12/MISO
D13/SCK
GND
Arduino
Uno
(Rev3)
R1
1kΩ
Q1
BC548
J1
Piezo Buzzer

**PROCEDURE:**

**STEP-1**

After writing the program you may save it with a file name of your choice (find File–>Save on menu bar of IDE)

**STEP-2**

You have to select the arduino board type in your IDE. I am using an Arduino Uno board. To choose the board, find Tools on menu bar. Choose the option "**Board"** – and select your correct arduino board.

**STEP-3**

The port number is assigned while installing the hardware driver of board. You may refer the tutorial on **Installing Arduino on Windows** to know how to find the port number of board. You can find the port number by accessing ***device manager*** on Windows. To select the right port, go to ***Tools–> Serial Port*** and select the port number.

There are two steps involved in loading the program from your PC to arduino board via the arduino IDE. First step is **compiling** and second step is called **burning**.

**STEP-4**

In the arduino IDE, compiling is called as "**verify**". So hit the verify button in your IDE (see the button with tick mark just below menu bar).

we are going to upload the verified program in arduino IDE to the arduino board. To do this, press the "**upload**" button (see the button with right arrow mark).

**PROGRAM:**

**void setup**() { // the setup function runs once when you press reset or power the board

pin Mode(13, OUTPUT);  // initialize digital pin 13 as an output.

Digital Write(13, HIGH); // turn the buzzer on (HIGH is the voltage level)

**}**

**CONCLUSION:**

# EXPERIMENT NO:-6

AIM OF THE EXPERIMENT:

To write an assembly language program for character based LCD Interface.

**APPARATUS REQUIRED:**
8051 trainer kit and interfacing kit.

**THEORY:**

- LCDs (Liquid Crystal Displays) are used for displaying status or parameters in embedded systems.
- 
- LCDs 16x2 is 16 pin device which has 8 data pins ($D_0$-$D_7$) and control pins (RS, RW, EN).The remaining 5 pins are for supply and backlight for the LCD.
- The control pins help us configure the LCD in command mode or data mode. They also help configure read mode or write mode also when to read or write.
- LCD 16x2 can be used in 4-bit mode or 8-bit mode depending on the requirement of the application. In order to use it we need to send certain commands to the LCD in command mode and once the LCD is configured according to our need, we can send the required data in data mode.

Here we are flashing display of good 51 on seven segment display on executing this program from address 2000H,"Good51" message flashes on the display of the kit.

**PROCEDURE:**

**STEP 1- STORAGE OF PROGRAM MACHINE CODES:**
Machine codes of the program to be executed, should be stored in their memory available on 8051 trainer kit as
**RESET→ EXMEM→** starting address of program (2000)**→NEXT→** Now enter all themachine codes, one after the other followed by the key **'NEXT'.**

**STEP 2-EXECUTION OF PROGRAM:**
The program can be executed as
**FILL→GO→** Starting address of the program (2000)**→FILL**

**STEP 3-VERIFICATION OF OUTPUT:**
Output of the executed program can be verified as

**RESET→ EXMEM→** type the memory address where result will be stored
**→NEXTPROGRAM:**

| **MEMORY ADDRESS** | **MACHINE CODES** | **LABLES** | **MNEMONICS** | **OPERANDS** | **COMMENTS** |
|---|---|---|---|---|---|
| 2000 | 90 20 2E | HERE | MOV | DPTR,#202E | ;Good 51 message |
| 2003 | 12 06 F7 | | LCALL | 06F7 | ;display routine |
| 2006 | 7B 00 | | MOV | R3,#0 | |
| 2008 | 7A 00 | LOOP2: | MOV | R2,#0 | |
| 200A | DA FE | LOOP1: | DJNZ | R2,200A | ;delay code |
| 200C | DB FA | | DJNZ | R3,2008 | |
| 200E | 90 20 34 | | MOV | DPTR,#2034 | ;blank message |
| 2011 | 12 06 F7 | | LCALL | 06F7 | ;display routine |
| 2014 | 7B 00 | | MOV | R3,#0 | |
| 2016 | 7A 00 | LOOP4; | MOV | R2,#0 | |
| 2018 | DA FE | LOOP3 | DJNZ | R2,2018 | ;delay code |
| 201A | DB FA | | DJNZL | R3,2016 | |
| 201C | 80 E2 | | SJMP | 2000 | |
| 202E | 41 03 03 85 49 9F | | DFB | 41,03,03,85,49,9F | Data Good 51 |
| 2034 | FF FF FF FF FF FF | | DFB | ,FF,FF FF,FF,FF,FF | BLANK DISPLAY |

**CONCLUSION:**

# EXPERIMENT NO:-7

AIM OF THE EXPERIMENT:

To interface stepper motor with 8051 parallel port and to vary speed of motor and direction of motor.

**EQUIPMENT REQUIRED:**

8051 microcontroller Stepper
motor interface board

**THEORY:**

A motor in which the motor is able to assume only discrete stationary angular position in a stepper motor. The motor occurs in a stepwise manner from one equilibrium position to next.

The motor under our consideration uses two phase scheme of operation. In this scheme any two adjacent stator winding are energized . The switching conditionfor the above said scheme shown in the table;

Clockwise

| A1 | B1 | A2 | B2 |
|---|---|---|---|
| 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |

| A1 | B1 | A2 | B2 |
|---|---|---|---|
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |

In order to verify the speed of the motor the values stored in the register R1, R2, R3 can be changed approximately.

Process:

1. Store the look up table address in DPTR.
2. Move the count table (04) to one of the register R0.
3. Load the control word for motor relation in all.
4. Push the address in DPTR into stack.
5. CALL the delay program.
6. Send the control word for motor relation to the external voice.

7. POP up the value in stack and increment it.
8. Decrement the count in R0 if 0 go to o the next. Stop wise else proceed to step-3.
9. Perform step-1 to step-0 respectively.

**PROCEDURE:**

**STEP 1- STORAGE OF PROGRAM MACHINE CODES:**

Machine codes of the program to be executed, should be stored in their memory available on 8051 trainer kit as

**RESET→ EXMEM→** starting address of program (2000)**→NEXT→** Now enter all themachine codes, one after the other followed by the key **'NEXT'.**

**STEP 2-EXECUTION OF PROGRAM:**

The program can be executed as

**FILL→GO→** Starting address of the program (2000)**→FILL**

**STEP 3-VERIFICATION OF OUTPUT:**

Output of the executed program can be verified as

**RESET→ EXMEM→** type the memory address where result will be stored **→NEXT**

**PROGRAMMING:-**

```
ORG 4100

START: MOV DPTR, #4500H

MOV R0, #0A

AGAIN: MOV A, @DPTR

        PUSH  DPH

        PUSH    DPL

        MOV DPTR, # FFCDH

        MOV, 0A #

        MOV R, # FFH

DLY1: MOV R3, #FFH

DLY:  DINZ R3, DLY
```

- DJNZ R2, DLY, n

- 

POP DPL

POP DPH

DJNZ R0, AGAIN

SJMP START

**CONCLUSION:**

- 

# EXPERIMENT NO.:-8

**AIM OF THE EXPERIMENT:**

Write a program to generate delay subroutine.

**EQUIPMENT REQUIRED:**

8051 Trainer kit

**THEORY:**

**Program for generating 1mS delay using 8051 timer.**

The program shown below can be used for generating 1mS delay and it is written as a subroutine so that you can call it anywhere in the program. Also you can put this in a loop for creating longer time delays (multiples of 1mS). Here Timer 0 of 8051 is used and it is operating in MODE1 (16 bit timer).

**PROCEDURE:**

**STEP 1- STORAGE OF PROGRAM MACHINE CODES:**

Machine codes of the program to be executed, should be stored in their memory available on 8051 trainer kit as

**RESET→EXMEM→** starting address of program (2000)**→NEXT→** Now enter all themachine codes, one after the other followed by the key **'NEXT'.**

**STEP 2-EXECUTION OF PROGRAM:**

The program can be executed as

**FILL→GO→** Starting address of the program (2000)**→FILL**

**STEP 3-VERIFICATION OF OUTPUT:**

Output of the executed program can be verified as

**RESET→ EXMEM→** type the memory address where result will be stored

**→NEXTPROGRAM:**

```
DELAY: MOV TMOD,#00000001B // Sets Timer 0 to MODE1 (16 bit timer). Timer 1
is not used
     MOV TH0,#0FCH // Loads TH0 register with FCH
     MOV TL0,#018H // Loads TL0 register with 18H
     SETB TR0 // Starts the Timer 0
HERE: JNB TF0, HERE // Loops here until TF0 is set (i.e.; until roll over)
    CLR TR0 // Stops Timer 0
    CLR TF0 // Clears TF0 flag
    RET
```

It is shown in the program below.

```
MAIN: MOV R6, #2D
LOOP: ACALL DELAY
    DJNZ R6, LOOP
    SJMP MAIN
DELAY: MOV TMOD, #00000001B
     MOV TH0, #0FCH
     MOV TL0, #018H
     SETB TR0
HERE: JNB TF0, HERE
    CLR TR0
    CLR TF0
    RET
```

**CONCLUSION:**

# EXPERIMENT NO.:-9

**AIM OF THE EXPERIMENT-**

Write simple program to interface traffic light systems.

**COMPONENTS REQUIRED:**

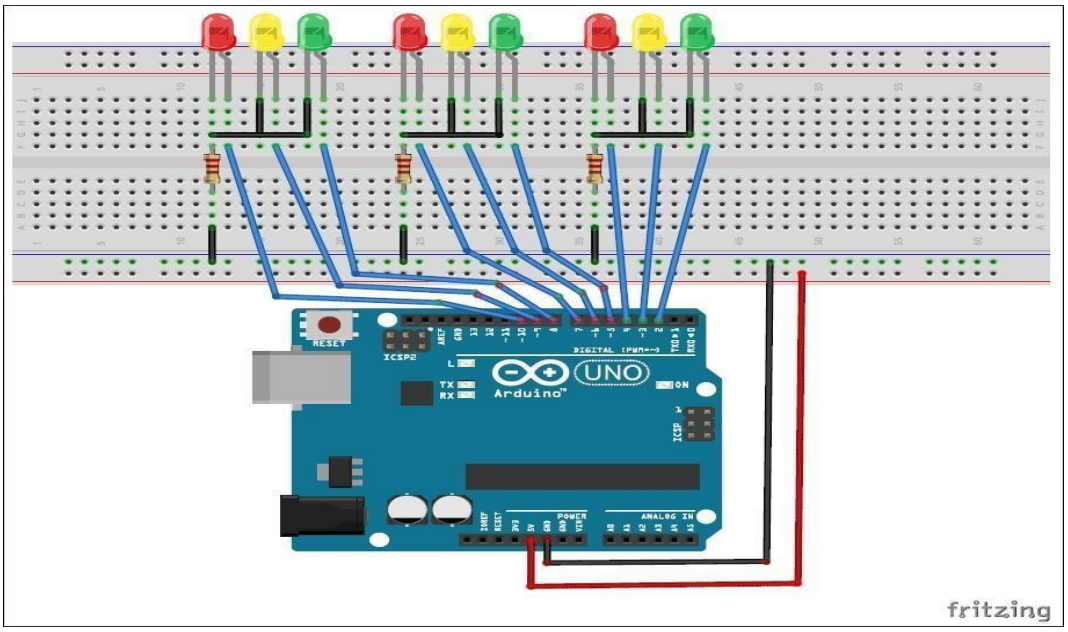
1. 3*Red LED Lights
2. 3*Green LED Lights
3. 3*Yellow LED Lights
4. 3*220ohm Resistors
5. Breadboard
6. Male To Male Connectors
7. Arduino Uno With Ide Cable

**THEORY:**

The code for this **Arduino Traffic Light Controller Project** is simple and can be easily understood. Here we have demonstrated Traffic lights for the 3 ways road and the code glows LED's on all the three sides in a particular sequence, in which the actual Traffic Lights works. Like, at a time, there will be two Red signals on any of the two sides and one Green light on the remaining side. And yellow light will also glow, for 1 second each time, in between transition from Red to Green, means first red light glows for 5 second then yellow light glows for 1 second and then finally green light will be turned on.

1. Connect the LEDs in the order as Red, Green, and Yellow in the breadboard.
2. Place the negative terminal of the LEDs in common and connect the 220ohm resistor in series.
3. Connect the connector wires accordingly.
4. Connect the other end of the wire to the Arduino Uno in the consecutive pins(2,3,4…10)
5. Power up the breadboard using the Arduino 5v and GND pin.

**CIRCUIT:**




**PROCEDURE:**

**STEP-1**

After writing the program you may save it with a file name of your choice (find File–>Save on menu bar of IDE)

**STEP-2**

You have to select the arduino board type in your IDE. I am using an Arduino Uno board. To choose the board, find Tools on menu bar. Choose the option "**Board"** – and select your correct arduino board.

**STEP-3**

The port number is assigned while installing the hardware driver of board. You may refer the tutorial on **Installing Arduino on Windows** to know how to find the port number of board. You can find the port number by accessing ***device manager*** on Windows. To select the right port, go to ***Tools–> Serial Port*** and select the port number.

There are two steps involved in loading the program from your PC to arduino board via the arduino IDE. First step is **compiling** and second step is called **burning**.

**STEP-4**

In the arduino IDE, compiling is called as "**verify**". So hit the verify button in your IDE (see the button with tick mark just below menu bar).

we are going to upload the verified program in arduino IDE to the arduino board. To do this, press the "**upload**" button (see the button with right arrow mark).

**PROGRAM:**

```
void setup() {
  // configure the output pins
  pin Mode(2,OUTPUT); pin
  Mode(3,OUTPUT); pin
  Mode(4,OUTPUT); pin
  Mode(5,OUTPUT); pin
  Mode(6,OUTPUT); pin
  Mode(7,OUTPUT); pin
  Mode(8,OUTPUT); pin
  Mode(9,OUTPUT); pin
  Mode(10,OUTPUT);
}

void loop()
{
  Digital Write(2,1); //enables the 1st set of
  signals digital Write(7,1); digital
  Write(10,1);digital Write(4,0); digital
  Write(3,0); digital Write(6,0); digital
  Write(8,0); digital Write(9,0); digital
  Write(5,0); delay(5000);

  Digital Write(3,1); //enables the yellow
  lights digital Write(6,1); digital Write(2,0);
  digital Write(7,0); delay(1000); digital
  Write(4,1); //enables the 2nd set of signals
  digital Write(5,1); digital Write(10,1);digital
  Write(2,0); digital Write(3,0);
```

```
  Digital Write(6,0); digital Write(8,0); digital
  Write(9,0); digital Write(7,0); delay(5000);

  Digital Write(9,1); //enables the yellow
  lights digital Write(6,1); digital
  Write(10,0);digital Write(5,0); digital
  Write(4,0); delay(1000);

  Digital Write(8,1); //enables the 3rd set of
  signals digital Write(4,1); digital
  Write(7,1);digital Write(2,0); digital
  Write(3,0); digital Write(5,0); digital
  Write(6,0);



  Digital Write(9,0); digital
  Write(10,0);delay(5000);


  Digital Write(9,1); //enables the yellow
  lights digital Write(3,1); digital
  Write(7,0);digital Write(8,0); digital
  Write(4,0); delay(1000);
}
```

**CONCLUSION:**

# EXPERIMENT NO.:-10

**AIM OF THE EXPERIMENT-**

Write a simple program for blinking of two separate LED.

**COMPONENTS REQUIRED-**

- Arduino Uno
- Breadboard
- LED (2nos.)

**THEORY:**

- Connect the positive terminal of led with 3 and 4 no. of Arduino Uno pin.
- Connect the negative terminal of led with ground pin of Arduino Uno.

**PROCEDURE:**

**STEP-1**

After writing the program you may save it with a file name of your choice (find File–>Save on menu bar of IDE)

**STEP-2**

You have to select the Arduino board type in your IDE. I am using an Arduino Uno board. To choose the board, find Tools on menu bar. Choose the option "**Board"** – and select your correct Arduino board.

**STEP-3**

The port number is assigned while installing the hardware driver of board. You may refer the tutorial on **Installing Arduino on Windows** to know how to find the port number of board. You can find the port number by accessing ***device manager*** on Windows. To select the right port, go to ***Tools–> Serial Port*** and select the port number.

There are two steps involved in loading the program from your PC to Arduino board via the Arduino IDE. First step is **compiling** and second step is called **burning**.

**STEP-4**

In the Arduino IDE, compiling is called as "**verify**". So hit the verify button in your IDE (see the button with tick mark just below menu bar).

We are going to upload the verified program in Arduino IDE to the Arduino board. To do this, press the "**upload**" button (see the button with right arrow mark).

**PROGRAM:**

```
int led Pins[ ] = {3, 4};        // an array of pin numbers to which LEDs are attached

int pin Count = 2;               // the number of pins (i.e. the length of the array)

int time Count = 500;            // delay
void setup( ) { for (int pin = 0; pin < pin

    Count;                  pin++)          {

    pin Mode(led Pins[pin], OUTPUT);

    }

}


void loop( ) {

  for (int pin = 0; pin < pin Count; pin++) {

      digital Write(led Pins[pin], HIGH); // turn the LED on (HIGH is the voltage level)

      delay(time Count); // wait for a second (milliseconds) digital Write(led Pins[pin],

      LOW);        // turn the LED off by making the voltage LOW delay(time Count);      //

      wait for a second (milliseconds

    }

}
```

**CONCLUSION:**