pt-NO: 11	Design solutions for smart Home using Arduino processor.
Date:	

objective:

To design a smart home system using an Arduino processor involves integrating various sensors and communication modules to automate and control different aspects of the home environment.

System and Software Tools Required.

1. ARM Evolution kit.
2. ARM development tools.
3. LCD
4. Arduino IDE

Algorithm.

Step-1: Initialize the serial communication, serve, pin modes and default states of LEDs and sensors.

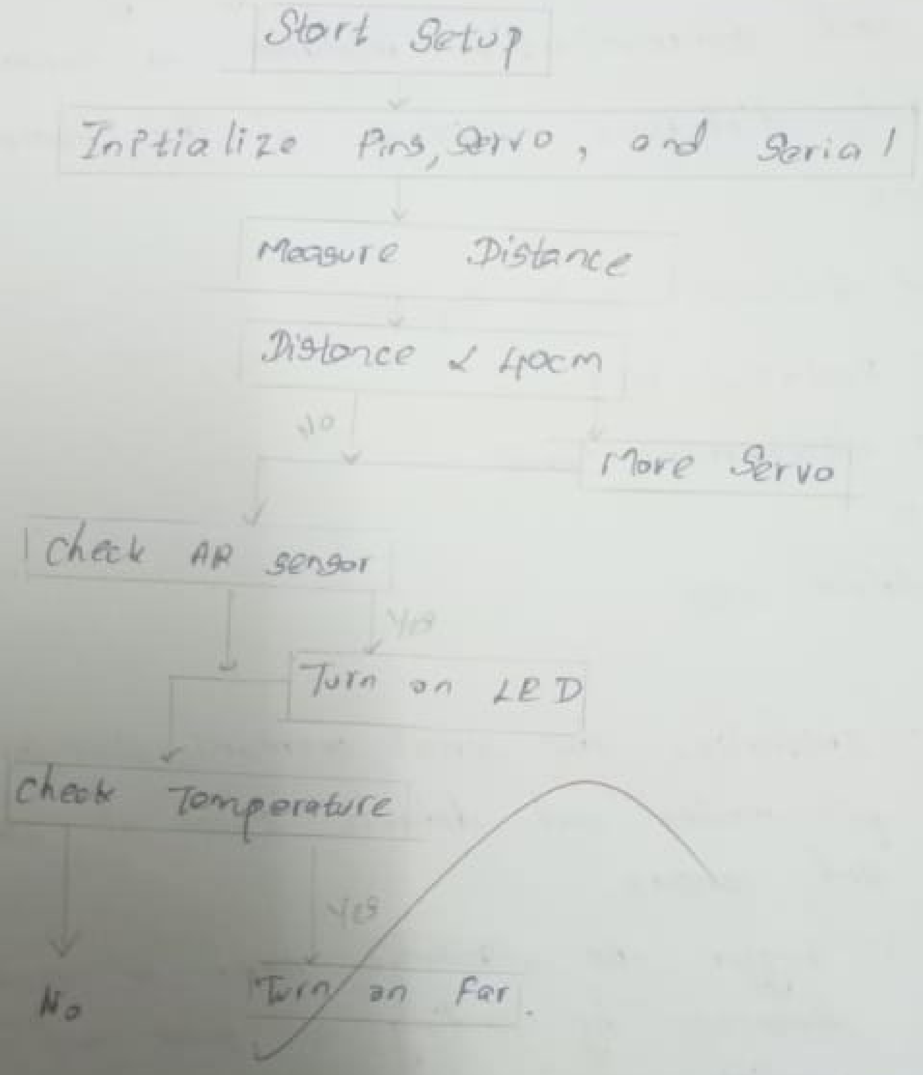
Step-2: Trigger the ultrasonic sensor to measure distance by sending a pulse and reading the echo

Step-3: Convert the time from the ultrasonic sensor to inches and centimeters.

Step - 5 : Monitor AR sensor, turn on an LED o
motion is detected.

Step - 6 : Read the temperature from the analog sensor and turn or a far (or LED) of the temperature exceeds 20°C

FLOWCHART :



ogram :

```
# include <serve.h>
const  int  pingPin = 7;
int  servoPin = 8;
servo   servo 1;
void   setup(){
      serial.begin (9600);
      serve.attach (servoPin);
      pinMode (2, input);
      pinMode (4,OUTPUT);
       pinMode ( 11, OUTPUT);
      pinMode (12, OUTPUT) ;
      pinMode (13,OUTPUT);
      pinMode (A0,INPUT);
      digitalWrite (2, LOW);
      digitalWrite (11,HIGH);
}
void  loop (){
     long  duration  inches , cm ;
     pinMode (pingPin , LOW);
     delay Microseconds (2);
     digitalWrite (pingPin, HIGH);
     delaymicroseconds (5);
     digitalWrite (pingPin , LOW) ;
     pinmode (Ping Pin , INPUT);
     duration  = pulseIn (ping Pin , HIGH);
     cm     =  microsecondsTocentimetersS(duration);
     servo.write (0);
if (cm<40){
   servo.write(90);
    delay (2000);
  }
else {
   Servo1.write(0);
}
int  pin = digitalRead(2);
if (pin == HIGH){
     digitalWrite(4, HIGH);
     delay(1000);
 }
  else if (pin ==LOW){
      digitalWrite(4,LOW);
  }
float  value = analogRead (A0);
float temp = value * 0.48 ;
serial.println ("temperature : ");
serial.println (temperature );
if (temperature >30){
      digitialWrite (12, HIGH);
  }
else{
   digitalWrite (12, LOW);
   }
}
long  microsecondsTocentimeters (long  microseconds){
    return microseconds  29/2 ;
 }
```

OGRAM :

```
#include <SPI.h>
#include <Ethernet.h>
byte mac [] = {0XDE, 0XAD, 0XBE, 0XEF, 0XFE,
                0XED}
IP/Address ip (192, 168, 1, 177);
IPAddress gateway (192, 168, 1, 1);
IPAddress subnet (255, 255, 0, 0);
EthernetServer server (23);
boolean gotAmessage = false.
void setup (){
  Ethernet. begin (mac, ip, gateway, subnet);
  server. begin ();
  serial. begin (9600);
}
 void loop () {
   Ethernet Client client = server. available ( );
   if (client){
     If (!gotAmessage) {
       serial. println ("We have a new client);
       client. println (" Hello, client! ");
       getAmessage = true;
     }
char thisChar = client.read();
server.write (thisChar);
 serial.print (thisChar);
  }
}
```

Exp. No: 09 Date:	Basic Programming Using ARM Processor using keil C

Objective:

To design and implement a program for interfacing an LED with an ARM processor and to verify the functionality of the LED by demonstrating various light patterns.

System and Software required:

1. ARM Evolvation kit
2. ARM Development tools
3. LED
4. KEIL C software.

Algorithm:

Step - 1 : Start the program.

Step - 2 : Enable input and output ports.

Step - 3 : Initialize times.

Step - 4 : Place the data for LED in data bus.

Step - 5 : Call delay.

Step - 6 : Rotate the data for LEDs to switch on next LED

Step - 7 : Use infinite loops for LEDs

Step - 8 : stop the program.

Expt. No: 10 Date:	Interfacing LCD Display with ARM using Keil C

Objective:

To design and implement a program for GPS navigation system to display the navigation data, directions and location information to users by interfacing an LCD display with ARM microcontrollers using Keil C.

System and Software tools required:

1. ARM Evaluation kit
2. ARM Development tools
3. LCD
4. Keil C Software.

Algorithm:

Step - 1 : Start the program.

Step - 2 : Enable input and output ports.

Step - 3 : Initialize timer.

Step - 4 : Place the data for LCD in the data bus.

Step - 5 : Call Delay.

Step - 6 : Rotate the data for the LCD to switch on next number

Algorithm

1. Set the three pin 13, Pin 9, pin6 as output pins.

2. Turn on LED on pin 13 by setting pin 13 High

3. Wait for 2 seconds and Turn off the LED on pin 13 by setting pin 13 to LOW.

4. Wait for 1 seconds and Turn on LED on pin9 by setting pin 9 to High.

5. Wait for 2 seconds.

6. Turn off LED on pin 9 by setting pin 9 to Low and wait for 1 second.

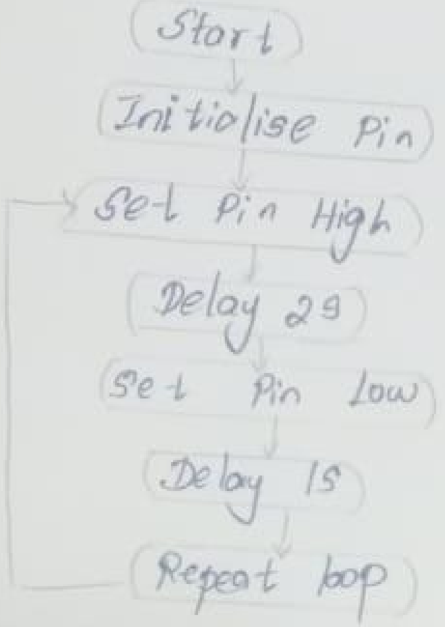
7. Turn on LED on pin6 by setting Pin6 to High.

8. Wait for 2 seconds. Turn off LED on pin6 by setting pin 6 to LOW.

9. Wait for 1 second.

10. Repeat the procedure for other pins.

OWCHART :



PROGRAM :

```
#C++ program.
void setup(){
    pinMode (13, Output);
    pin Mode (9, Output);
    pinmode (6, Output);
}
Void loop(){
     digitalWrite (13, High);
      delay (2000);
      digitalWrite (13,Low);
      delay (1000);
      digitalWrite (9, High);
       delay (2000);
       digitalWrite (6, High);
       delay (2000);
```

Exp No: 08

Date:

DESIGN A SIMPLE CHAT SERVER USING ARDUINO

OBJECTIVE:

To Design and implement a basic chat server system using Arduino microcontrollers to establish a simple communication protocol over a serial interface, allowing for real-time text messaging between multiple Arduino-based chat clients.

SYSTEM AND SOFTWARE TOOLS REQUIRED:

1. Arduino Board.
2. Simulation Software.

ALGORITHM:

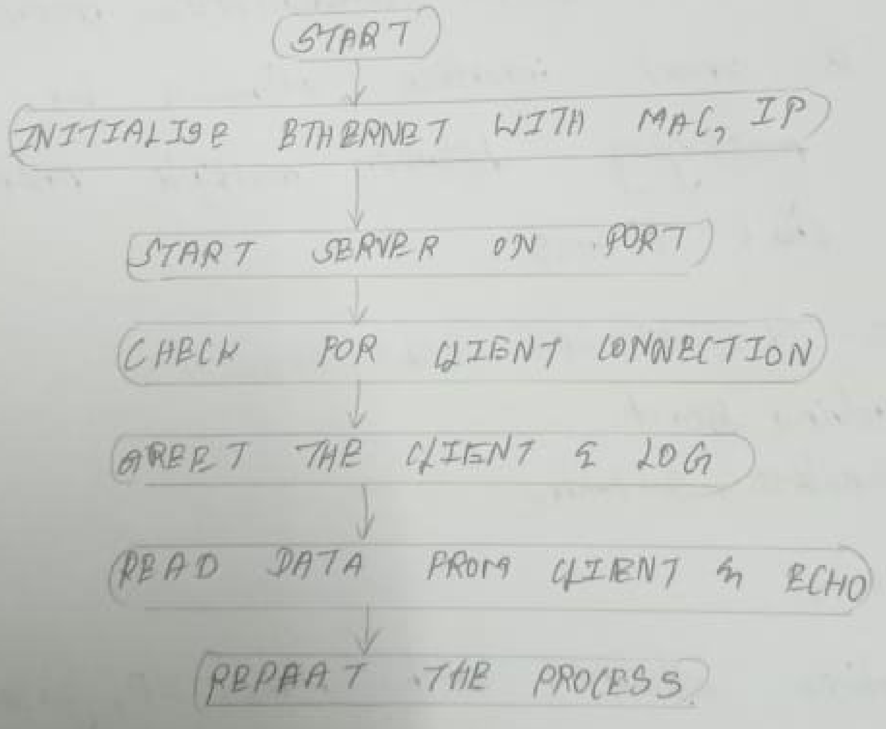
1. Initialise ethernet with MAC, IP, gateway, and subnet using Ethernet.begin().
2. Start server on port 23 using server.begin().
3. Check for client connections with server.available().
4. Great the new client connections with "Hello, client!" and log the connection.

5. Read incoming data using client.read().

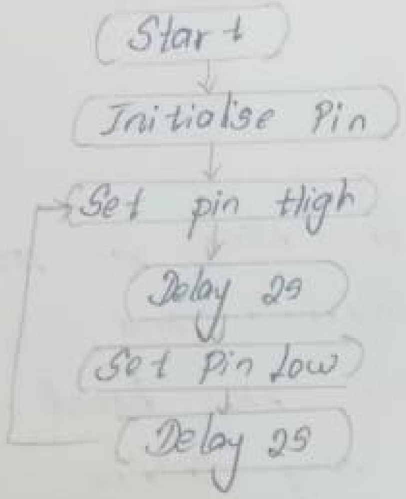
6. Echo Data back to the client and display it in the Serial.

7. Repeat the process continuously.

FLOWCHART:


START
INITIALISE ETHERNET WITH MAC, IP
START SERVER ON PORT
CHECK FOR CLIENT CONNECTION
GREET THE CLIENT & LOG
READ DATA FROM CLIENT & ECHO
REPEAT THE PROCESS

FLOWCHART:



PROGRAM:

C++ Program:

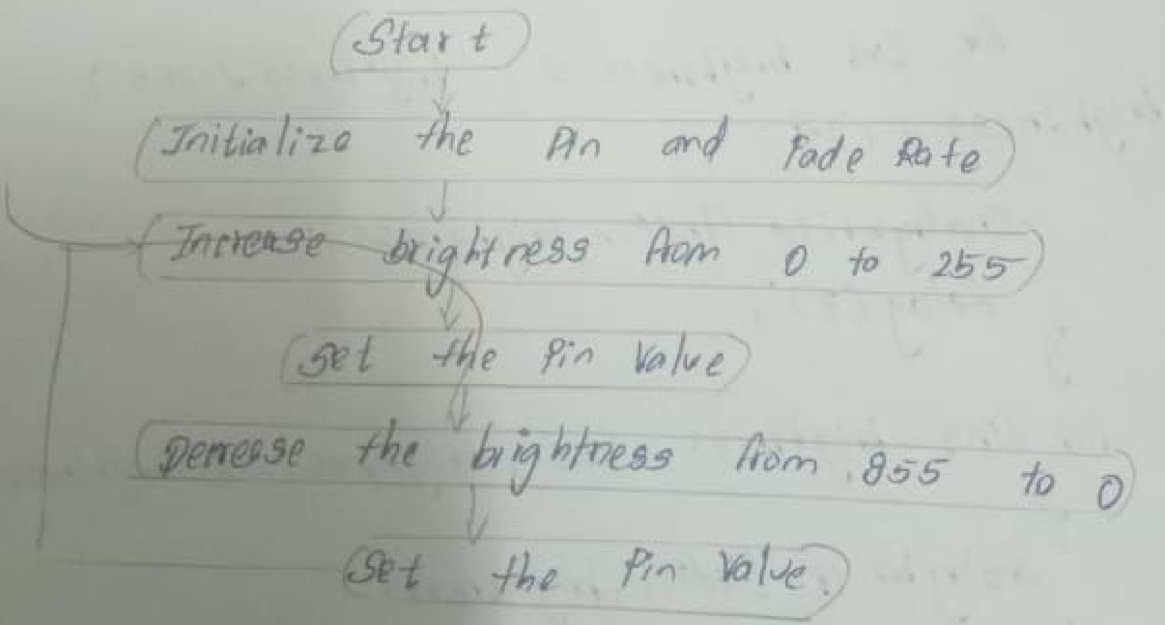
```
void setup() {
    PinMode (B,Output):
}
void loop() {
        digitalWrite (B,High);
        delay (2000);
        digitialWrite (B,Low);
        delay (1000);
    }
```

ED FADE :

Algorithm :-

1. Initialize the Pin (ie 13 & 8)
2. Icrease the brightness of Pin 13 gradually.
3. Decrease the brightness of Pin 13 gradually.
4. Increase Pin 8 brightness gradually.
5. Decrease the Pin 8 brightness gradually..

Flowchart ;


Start
Initialize the Pin and Fade Rate
Increase brightness from 0 to 255
Set the Pin Value
Decrease the brightness from 255 to 0
Set the Pin Value.

Program :

```
 # C++ program
 const int ledPin = 9 ;
 const int fadeRate = 5 ;
 void setup ( ){
       pinMode (ledPin , output);
       pinMode (8, output);
  }
 void loop (){
       for (int brightness =0 ; brightness <=255 ;
brightness ++){
       analogwrite (ledPin, brightness );
       delay (3);
   }
  for (int brightness =255 ; brightness>=0 ; brightness +-)
   {
    analogWrite (ledPin, brightness);
    delay (3);
  }
 // same program for pin 8
}
```

Exp.No : 7

Date :

DESIGN OF A TRAFFIC LIGHT CONTROLLER USING ADUINO

Objective

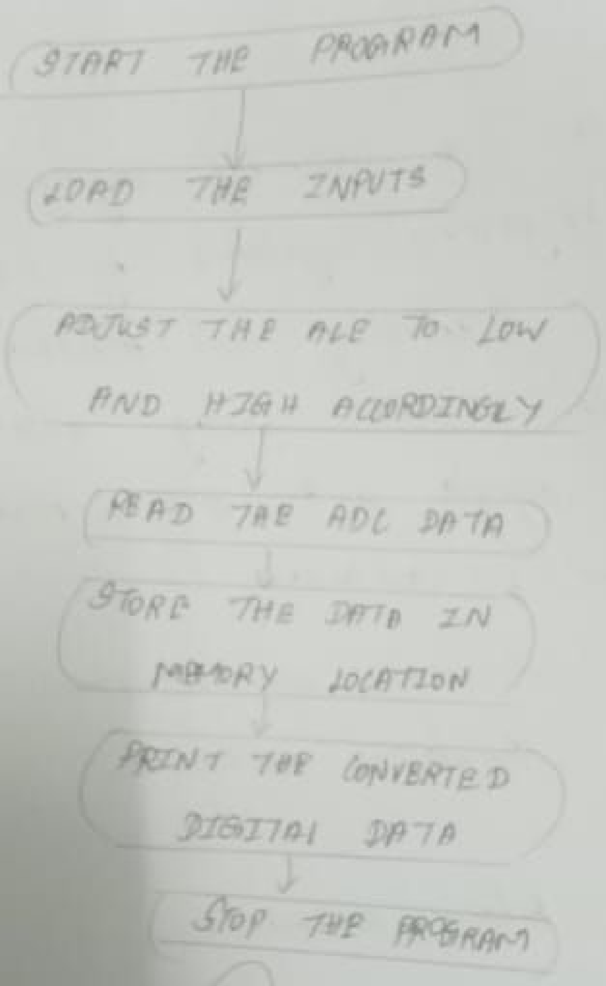
To design and implement a traffic light control system using an Arduino microcontroller that manage the timing and sequencing of traffic lights for different directions, including handling pedestrian crossing and traffic flow optimization.

System and Software Required :

1. Arduino Board
2. Simulation Software.

A traffic light controller using Arduino is a popular project that demonstrates the use of Arduino microcontrollers for controlling traffic lights.

FLOWCHART :


START THE PROGRAM
LOAD THE INPUTS
ADJUST THE ALE TO LOW AND HIGH ACCORDINGLY
READ THE ADC DATA
STORE THE DATA IN MEMORY LOCATION
PRINT THE CONVERTED DIGITAL DATA
STOP THE PROGRAM

GRAM:

ADDRESS	OPCODE	MNEMONICS
4100	90 FF C8	MOV DPTR, #FFC8
4103	74 10	MOV A, #10
4105	F0	MOVX @DPTR. A
4106	74 18	MOV A, #18
4108	F0	MOVX @DPTR, A
4109	90	MOV DPTR, #FFD0
410C	74 01	MOV A, #01
410E	F0	MOVX @DPTR, A
410F	74 00	MOV A, #00
4111	F0	MOVX @DPTR, A
4112	10 FF C0	MOV DPTR, #FFD8
4115	E0	MOVX. A, @DPTR
4116	30 E0 FC	JNB E0, 4115
4119	90 FF C0	MOV DPTR, #FFC0
411C	E0	MOVX A, @DPTR
4100	F0	MOVX @DPTR, A
4121	80 FE	SJMP 4121

INPUT : (In ACD kit)

8	4	2	1		8	4	2	1
X		X		▨				X

OUTPUT : (In 8051 Microcontrollers)

Memory Location	Output Data
4122	A1

Exp. No: 6 BASIC PROGRAM WITH ARDINO KIT

Date:

Objective:

To write a program to blink van LED on and off at regular intervals and control and brightness of an LED based on ambient height levels using Arduino.

System and Software tool Required:

1. Arduino board
2. Arduino IDE

Algorithm:

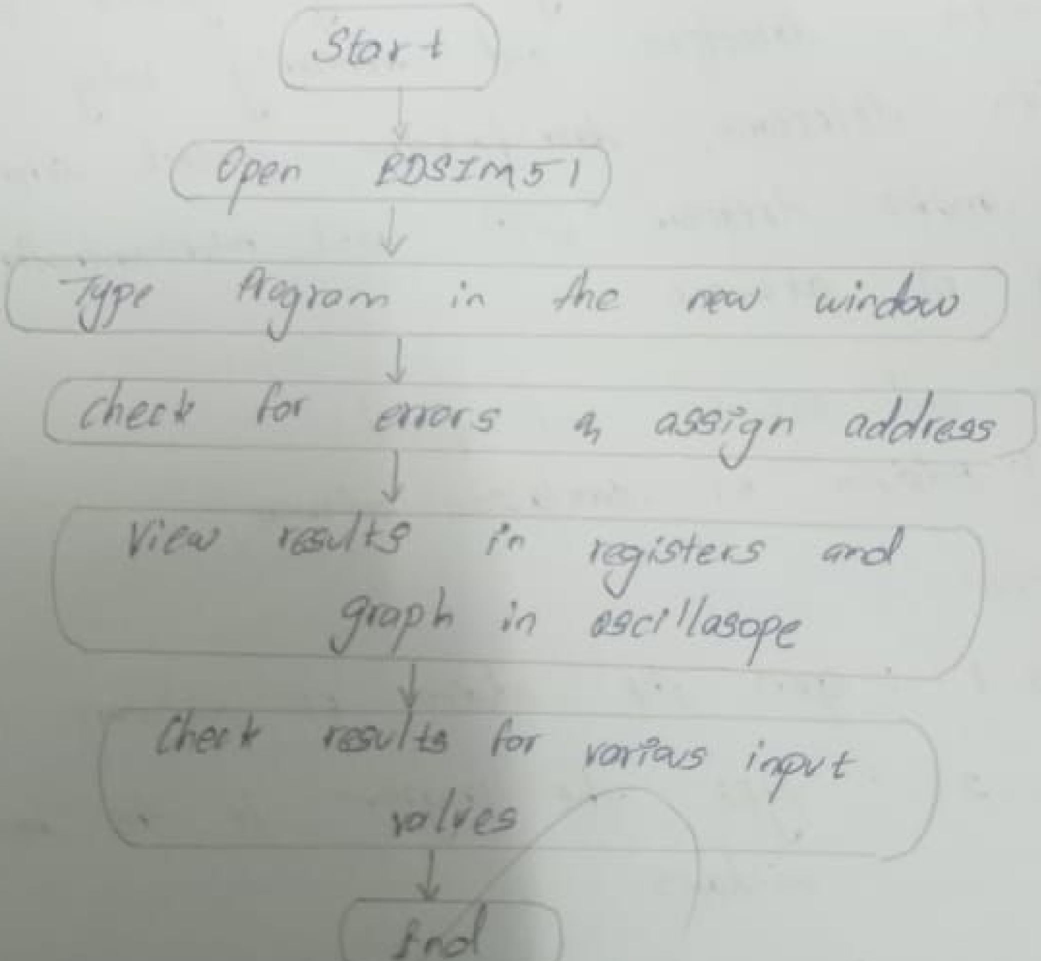
Basic LED Blinking:

1. Initialise the pin modes as output.
2. Set the Pin to high.
3. Wait for 2 seconds
4. Set the Pin to low.
5. Wait for 1 second.
6. Repeat the Procedure loop.

Step -4 : View the reset in registers and graph in oscillascope.

Step - 5 : For various values of inputs check the results.

Flowchart :


Start
Open RDSIM51
Type Program in the new window
check for errors & assign address
View results in registers and graph in oscillasope
Check results for various input values
End

gram :

1) Digital to Analog converters

MEMORY	LABEL	MNEMONICS	COMMENT
0000		CLR P0,7	
0002	START	MOV P1, #50H	Lead initial value for top of square wave.
0005		LCALL DELAY	Call the delay
0008		MOV P1, #0BBH	Load the value of TON
000B		LCALL DELAY	Call the delay
000E		SJMP START	Repeat
0010	DELAY	MOV R1, #45H	Move delay value to R1
0012	L2	DJNZ R1, R2	Decrement & Repeat
0014		RET	Return to main program

ii) Echoring the switches on the LEDS

Memory	Label	Mnemonics	Comment
0000	START	MOV P_1, P_2	More data on part 2 to port 1
000 3		JMP START	Repeat.

XP.NO: 05 | INTERFACING 8051 WITH ADC

DATE:

OBJECTIVE:

To write an assembly language program for developing irrigation systems by interfacing systems by interfacing soil moisture sensors with an ADC to convert analog moisture levels into digital signals with 8051 microcontroller using Edsim 51 software.

SYSTEM AND SOFTWARE TOOL REQUIRED:

1. Edsim 51 simulation software.

PROCEDURE:

1. Open the Edsim software.
2. Type the program in the new window.
3. Click Assemble, simulator will check for errors and assigns address for each byte of instruction.
4. Adjust the potentiometer & note down the corresponding digital output in ADC selection of software.
5. For various values of inputs in potiometer & check the results

xp No: 3	STEPPER MOTOR CONTROL USING 8086
Date:	MICROPROCESSOR

OBJECTIVE:

To write an assembly language program using stepper motor for controlling the speed and position of the conveyer belt in manufacturing or package industries for efficient item transparent and sorting using 8086 emulator.

SYSTEM AND SOFTWARE TOOLS REQUIRED:

1. 8086 Emulator (EMU8086)

PROCEDURE:

1. Open the emulator window and create a new .asm file.

2. Type the program for stepper motor control

3. Click the emulator icon, a new dialog box will open.

4. Click the run icon, program will now get executed.

5 Simulation window of stepper motor will appear now and rotation of stepper motor can be observed there.

ADDRESS	OPCODE	MNEMONICS
1100	B0 80	MOV AL,80
1102	BA 36 FF	MOV DX,FF36
1105	EE	OUT DX,AL
1106	BE 00 12	START: MOV SI,1200
1109	B3 04	MOV BL,04
110B	8A 04	REPEAT: MOV AL,[SI]
110D	BA 30 FF	MOV DX, FF30
1110	EE	OUT DX, AL
1111	E8 07 00	CALL DELAY
1114	46	INC SI
1115	FE CB	DEC BL
1117	75 F2	JNE REPEAT
1119	EB EB	JMP START
111B	B9 03 09	DELAY: MOV CX,0903
111E	49	LOOP : DEC CX
111F	75 FD	JNE LOOP
1121	C3	RET

OUTPUT :

Clockwise - direction :

Address :

```
2100   - 03
2101   - 06
2102   - 0C
2103   - 09
```

Anti -clockwise direction :-

```
2100   - 09
2101   - 0C
2102   - 06
2103   - 03
```

RESULT :

Thus, the program using stepper motor for controlling the speed and position of the conveyor belt in manufacturing or packaging industries was executed using 8086 emulator xpt. No : 04 | Sensor Interfacing using 8051

Date : | Microcontroller

Objective :

To write an assembly language program for developing security systems for intrusion detection and monitoring using motion detectors, door / window contact sensor and smoke detector with 8051 microcontroller using 51 software

System and software tool required :

1. Edsium 51 simulation software.

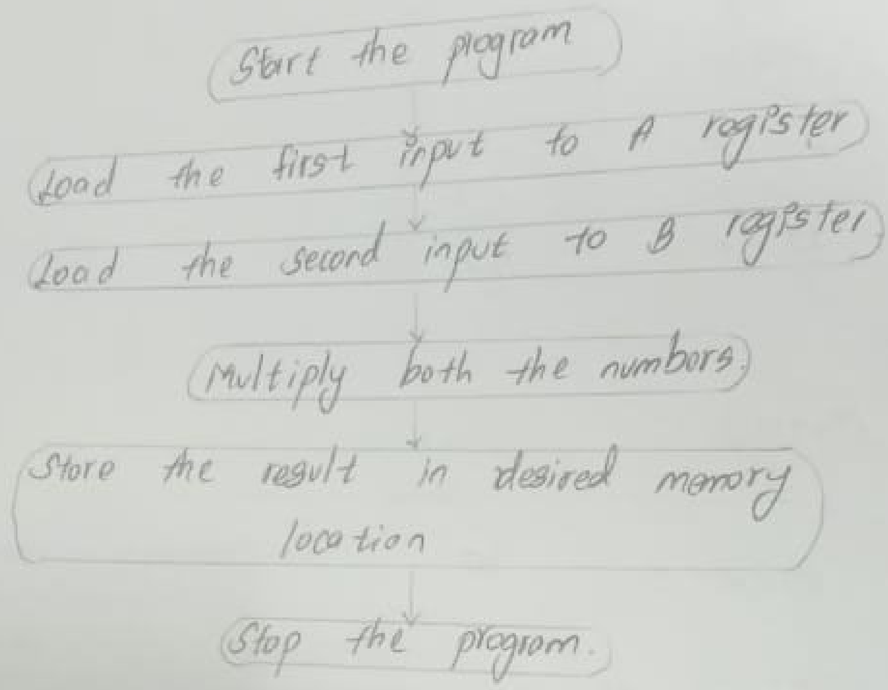
Algorithm :

Step - 1 : open the edsim 51 software.

step - 2 : Type the program in the new windows.

step - 3 : Click assemble, simulation will check for errors and assigns address for each byte of instruction.

FLOWCHART:


Start the program
Load the first input to A register
Load the second input to B register
Multiply both the numbers.
Store the result in desired memory location
Stop the program.

PROGRAM: MULTIPLICATION

ADDRESS	MNEMONICS	COMMENTS
8500	MOV A, #06	Move 8 bit fast data to A register
8502	MOV F0, #03	Move 8 bit second data to B register,
8505	MUL AB	multiply the data.
8506	MOV DPTR, #8600	Initialise memory pointer,
8509	MOVX, @DPTR, A	Store the result.
850A	INC DPTR	Increment memory pointer,
850B	MOV A, F0	Get the pakaged BCD
850D	MOVX, @DPTR,A	Store result
850E	Here Sjmp Here	End.

INPUT:

Memory location	Input Data
8500	06h
8502	03h

OUTPUT:

Memory location	Output Data
8600	12

DIVISION OF TWO NUMBERS:

ALGORITHM:

1. Start the program.

2. Load the 1st Number to A register.

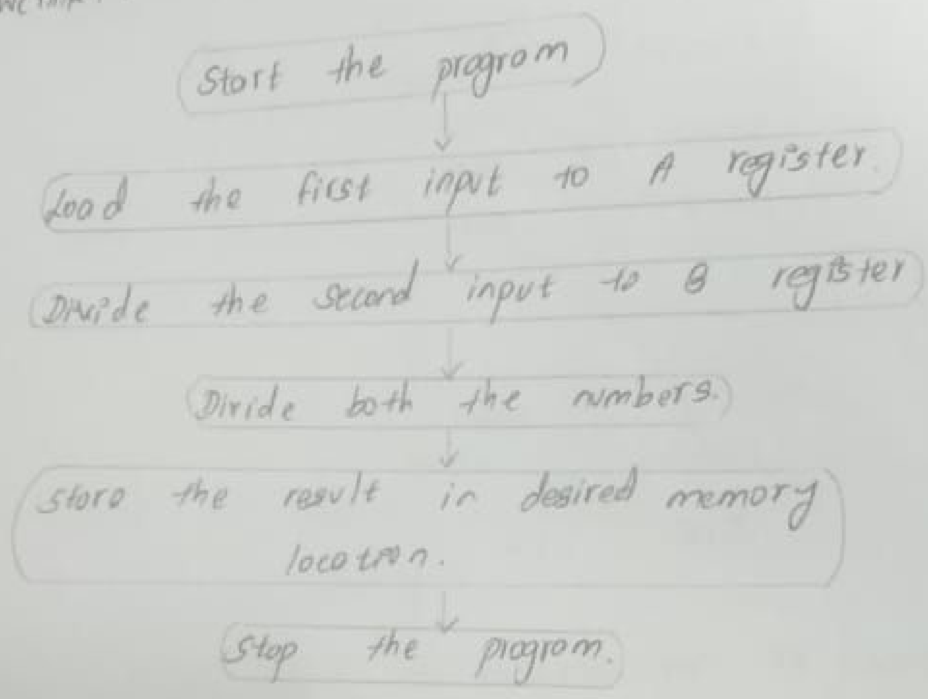
3 Load the 2nd Number to B register.

4. Divide both the Numbers.

5 Store the result in memory location.

6 Halt the program.

FLOWCHART :


Start the program
Load the first input to A register.
Divide the second input to B register
Divide both the numbers.
Store the result in desired memory location.
Stop the program.

PROGRAM : DIVISION

ADDRESS	MNEMONICS	COMMENTS
8500	MOV A, #09	Move 8bit first data to A register.
8502	MOV F0, #03	Move 8 bit second data B register.
8505	DIV AB	Divide the data
8506	MOV DPTR, A 8600	Initialize memory pointer.
8509	MOVX, @DPTR, A	Store the result.
850A	INC , DPTR	Increment memory pointer.
850B	MOV A0, F0	Get the packed BCD No.
850F	HERE SJMP HERE	End

Memory Location	Input Data
8500	09h
8502	03h

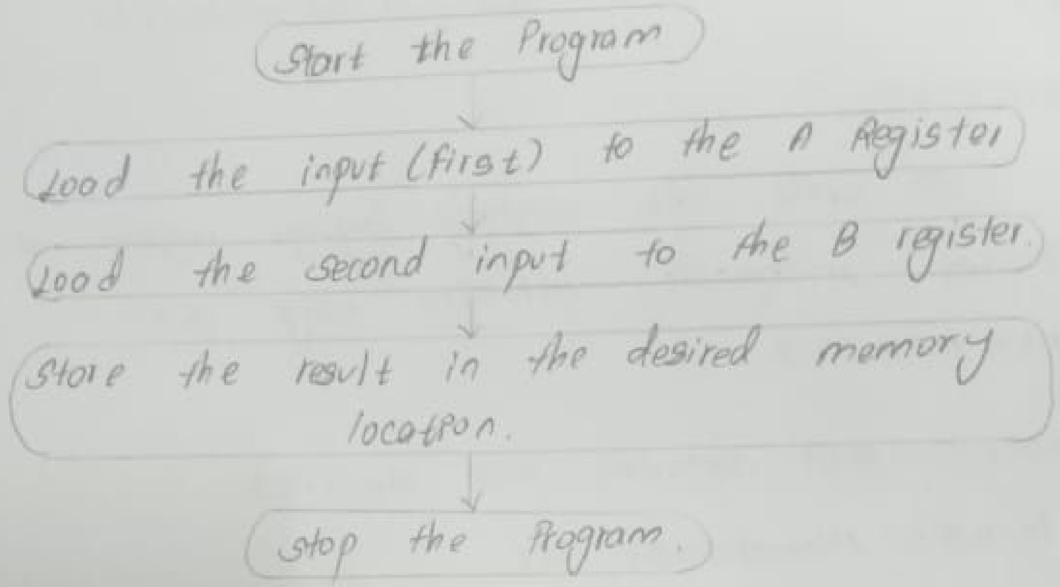
OUTPUT :

Memory Location	Output Data
8600	3

RESULT :

Thus the program for addition, subtraction, multiplication & division of given data was executed & verified.

FLOWCHART :


Start the Program
Load the input (first) to the A Register
Load the second input to the B register.
Store the result in the desired memory location.
stop the Program.

PROGRAM : ADDITION :

ADDRESS	MNEMONICS	COMMENTS
8500	MOV A, #13	Move 8 bit first data to A register
8502	ADD A, #14	Add 8 bit second data with A register.
8504	MOV DPTR, #8500	Initialize memory pointer.
8507	MOVX ,@DPTR, A	Store the Result
8508	Here SJMP HER Here	End.

PUT:

Memory Location	Input Data
8500	Bh
8502	14h

OUTPUT:

Memory Location	Output Data
8500	27

SUBTRACTION OF TWO NUMBERS:

ALGORITHM:

1. Start the program.

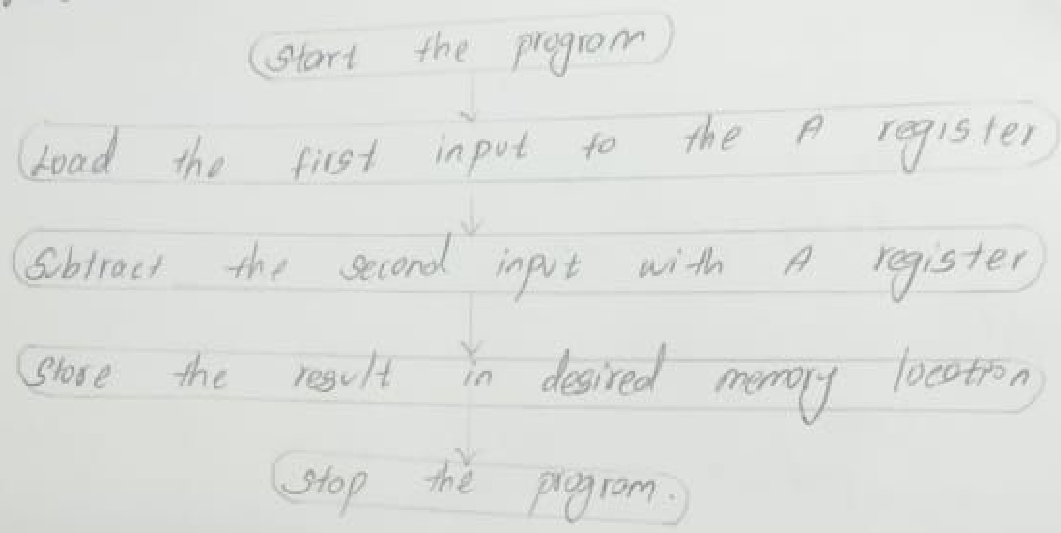
2. Load the 1st Number to A register.

3 Subtract 8 bit second data from A register.

4 Store the result in memory location.

5 Halt the program.

FLOWCHART:



Start the program

Load the first input to the A register

Subtract the second input with A register

Store the result in desired memory location

Stop the program.

PROGRAM : SUBTRACTION

ADDRESS	MNEMONICS	COMMENTS
8500	MOV A, #20	Move 8 bit first data to A register.
8502	SUB A, #20	Subtract 8 bit second data from A
8504	MOV DPTR, #8500	Initialize memory pointer.
8507	MOV X, @DPTR	Store the result.
8508	Here SJMP Here	End.

NPUT :

Memory Location	Input Data
85 00	20h
85 02	10h

OUTPUT

Memory Location	Output Data
8500	10

MULTIPLICATION OF TWO NUMBERS :

ALGORITHM :

1. Start the program.
2. Load the 1st Number to A register.
3. Load the 2nd Number to B register.
4. Multiply both the numbers.
5. Store the result in memory location.
6. Halt the program.

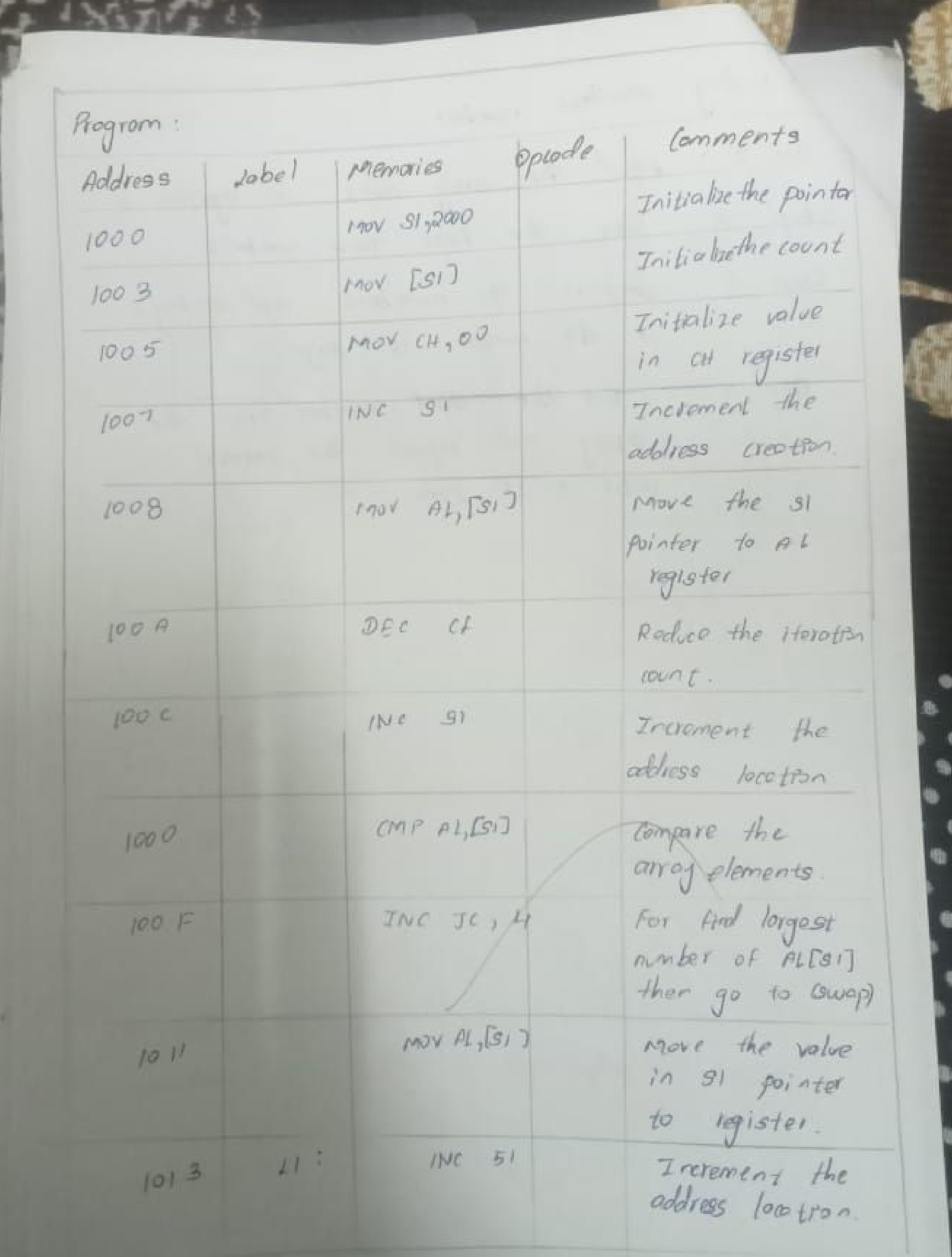
ii) Finding smallest number

Step 1 : Load the array count in register c

Step 2 : Get the first two numbers.

Step 3 : Compare the numbers and exchange if the number is large

Step 5 : Get the third number from the array and repeat the process until c is zero.



Program :

Address	Label	Memories	Opcode	Comments
1000		MOV SI,2000		Initialize the pointer
1003		MOV [SI]		Initialize the count
1005		MOV CH,00		Initialize value in CH register
1007		INC SI		Increment the address creation.
1008		MOV AL,[SI]		Move the SI pointer to AL register
100A		DEC CH		Reduce the iteration count.
100C		INC SI		Increment the address location
100D		CMP AL,[SI]		Compare the array elements.
100F		INC JC, L1		For find largest number of AL[SI] then go to (swap)
1011		MOV AL,[SI]		Move the value in SI pointer to register.
1013	L1 :	INC SI		Increment the address location.

Address	Label	Memories	Opcode	Comments
1014		Loop L2		Go to location L2 [1000] and repeat the loop
1016		MOV[2100], AL		move the result in AL to a location and store.
101A		HLT		Stop the program.

Input :

Memory Location	Input Data
CL	03h
2000	03h
2001	06h
2002	01h

Output [smallest]

Memory location	Output Data
2100	01

Output [largest]

Memory Location	Output Data.
2100	06

xp. No: 2 Date :	SIMPLE ARITHMETIC OPERATIONS USING 8051 MICROCONTROLLERS

OBJECTIVE:

To write the assembly language program to perform arithmetic operations using 8051 microprocessors.

SYSTEM AND SOFTWARE TOOL REQUIRED:

1. 8051 Microprocessor kit.

2. Key board

ADDITION OF TWO NUMBERS :

ALGORITHM:

1. Start the program

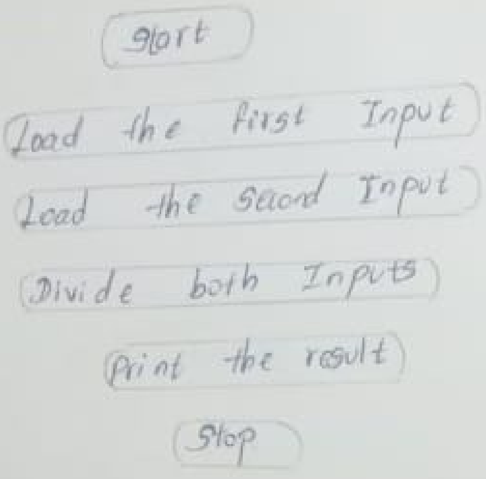
2. Load the first number to A register.

3. Add 8 bit second data to A register.

4. Store the Result in Memory Location.

5. Halt the program.

FLOW CHART :



PROGRAM : DIVISION

ADDRESS	MNEMONICS	COMMENT
1100	MOV DX,0000	Move the 16 bit data 000H
1103	MOV AX, 1212	Move the 1st no. to AX register
1106	MOV BX, 02	Move the 2nd no. to BX
1109	DIV BX	Divide both the numbers
110B	MOV DI, 1520	Transfer the o/p to DI
110E	MOV [DI], AL	Transfer the content of AL
1110	INC DI	Increment DI register
1111	MOV[DI], AH	Transfer content of AH to 1521
1113	INC DI	Increment DI register
1114	HLT	Stop the program.

INPUT

MEMORY LOCATION	INPUT DATA
AX	1000
BX	0010

OUTPUT

MEMORY LOCATION	OPUT DATA
1200	00

Exp No: 1b	Array Programming - Finding the largest and smallest number
Date:	

Objective:

To write on Assembly language Program for finding the smallest and largest element in an array using 8086 trainer kit.

System and software Tool required:

1) 8086 Microprocessor kit
2) keyboard

Algorithm:

1) Finding largest number

Step - 1: Load the array count in a register C

Step - 2: Get the first two numbers.

Step - 3: Compare the numbers and exchange if the number is small.

Step - 4: Get the third number from the array and repeat the process until C is 0.

FLOWCHART :

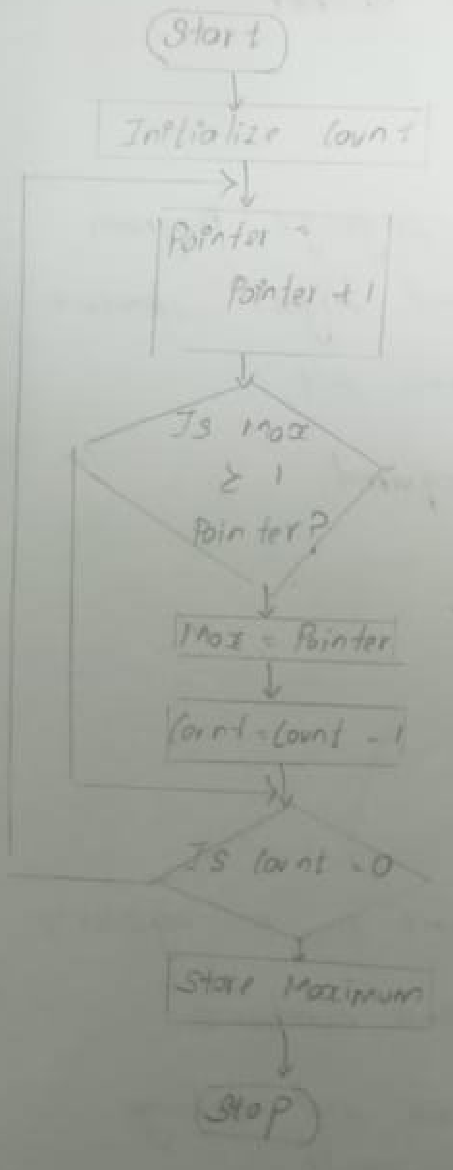


Fig finding the largest number

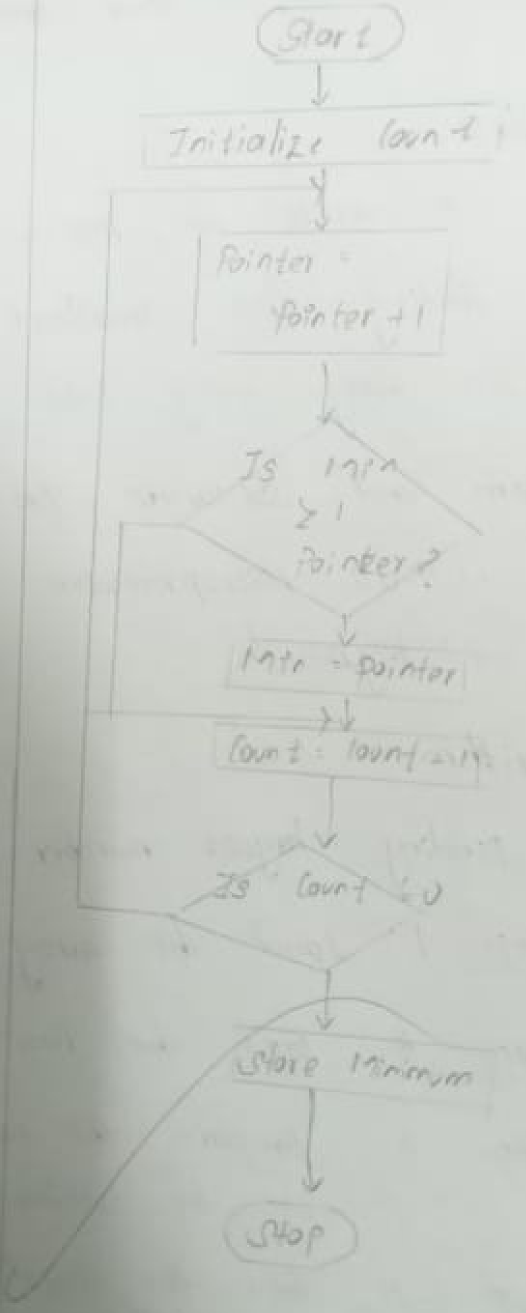
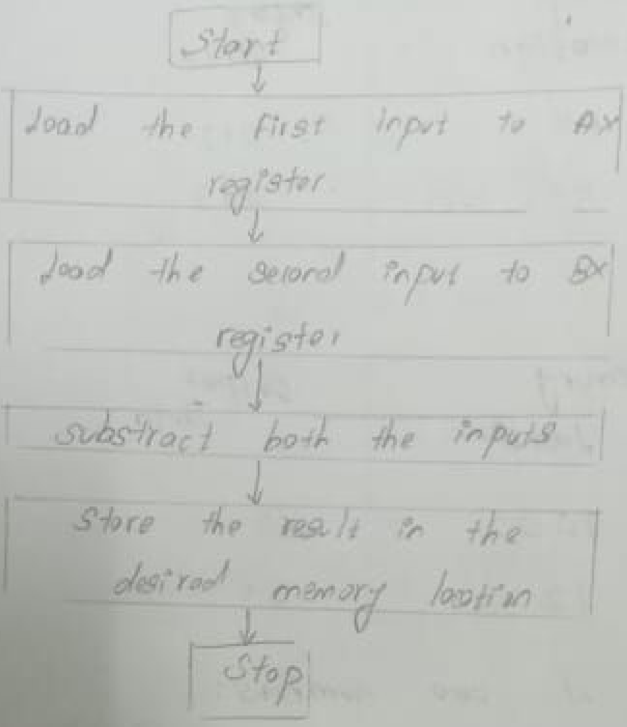


Fig finding the smallest number

FLOWCHART:


Start
load the first input to AX register
load the second input to BX register
substract both the inputs
Store the result in the desired memory location
Stop

Program: Subtraction

Address	Memorics	Comments.
1100	MOV AX, 1212	More 1st no. to AX
1103	MOV BX, 1313	More 2nd no to BX
1106	SUB AX, BX	substract both number
110B	MOV [1200], AX	Store result in memory locotion.
101A	HLT	Stop the program.

INPUT

MEMORY LOCATION	INPUT DATA
AX	1212 h
BX	1313 h

OUTPUT

MEMORY LOCATION	OUTPUT
1200	FF
1201	FB

Multiplication of two numbers:

Algorithm :

1. Start the program.

2. Load the 1st number to AX register.

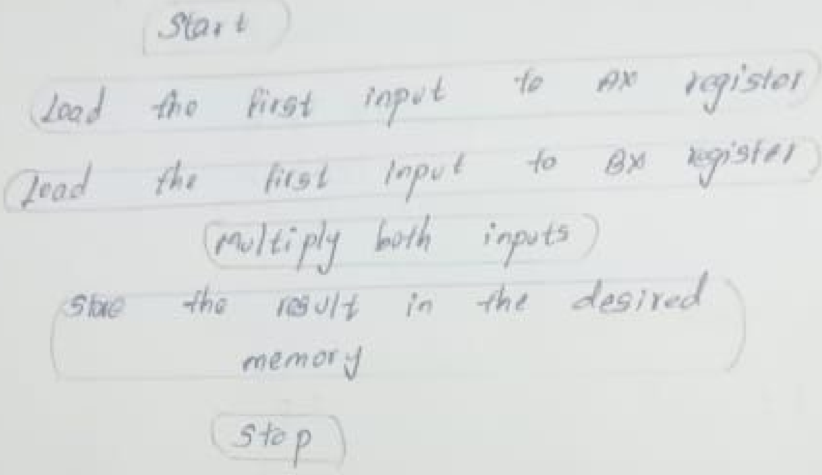
3. Load the 2nd number to BX register.

4. Multiply both number.

5. Store the result in memory location.

6. Halt the program

FLOWCHART


Start
Load the first input to AX registor
Load the first input to BX registor
Multiply both inputs
Store the result in the desired memory
Stop

Program : Multiplication

ADDRESS	MNEMONICS	COMMENTS
1100	MOV DX, 0000	Move the 16 bit data 000H to DX
1103	MOV AX, 0002	Move 1st no. to AX register
1106	MOV BX, 02	Move the 2nd no. to BX register
1109	MUL BX	Multiply both no.
110B	MOV DI, 1520	Transfer the output location
110E	MOV [DI], AL	Content of AL register to 1520
1110	INC DI	Content increment DI register
1111	MOV [DI], AH	Content of AH register to 1521
1113	INC DI	Increment DI register
1114	MOV [DI], DX	Content of DX to 1522
1116	HLT	Stop the program

INPUT

Memory location	Input data
AX	1212
BX	1313

OUTPUT

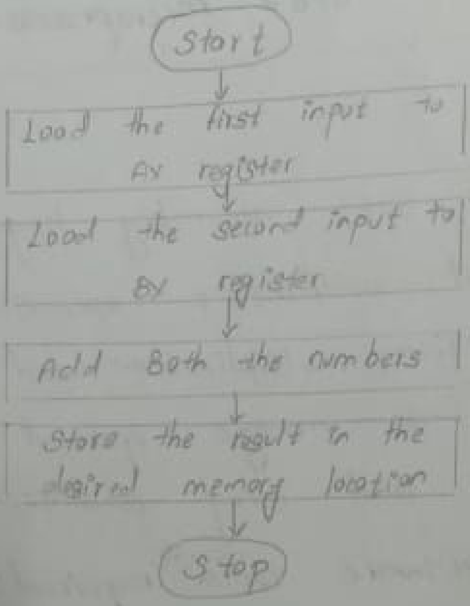
Memory location	Output data
[redacted] 1200	44
1201	84

Divison of two numbers:

ALGORITHM:

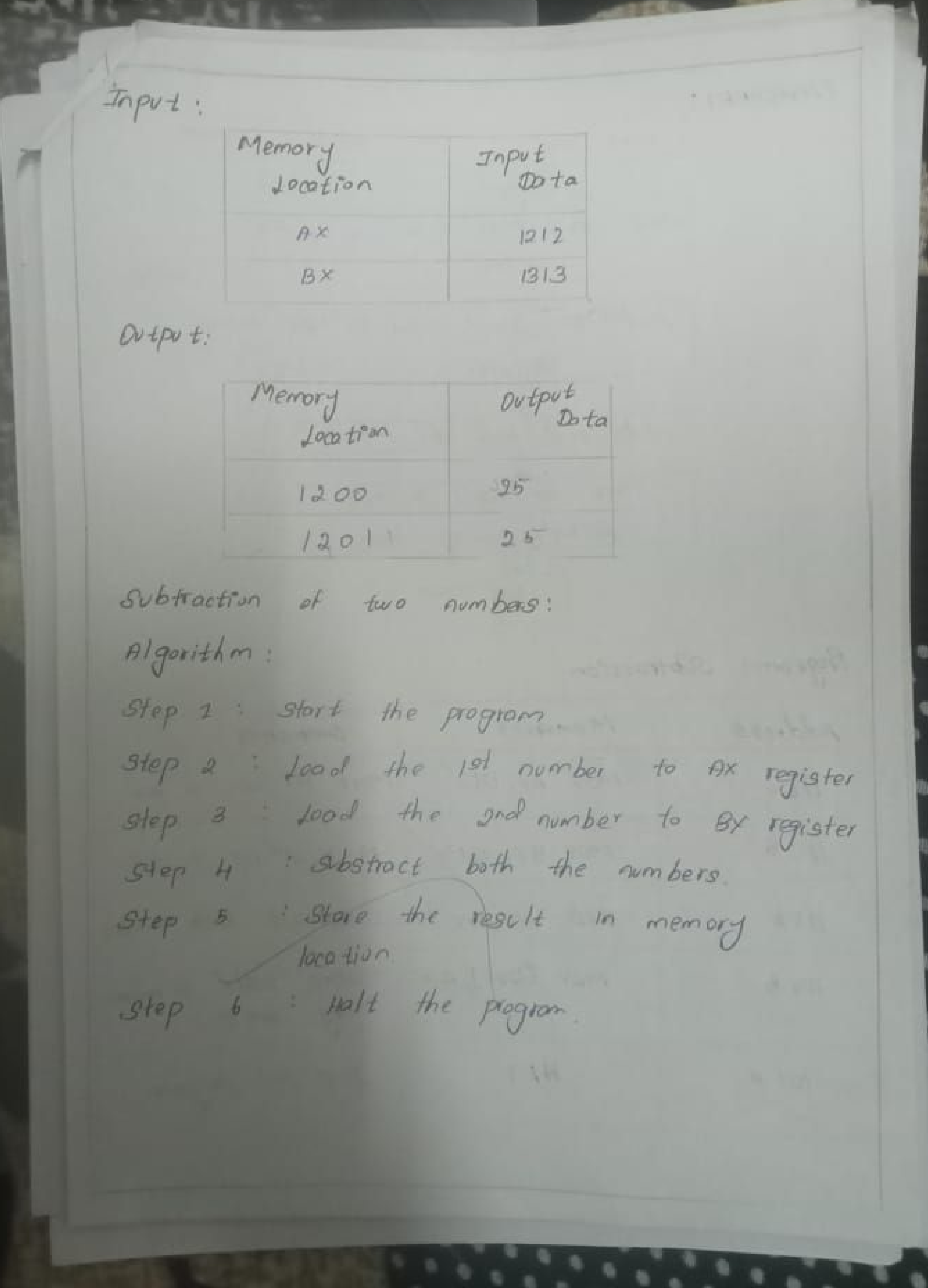
1. Start the program
2. Load the 1st number to AX register
3. Load the 2nd number to BX register.
4. Divide both numbers
5. Store the result
6. Halt the program.

FLOWCHART :


Start
Load the first input to AX register
Load the second input to BX register
Add Both the numbers
Store the result in the desired memory location
Stop

Program : Addition.

Address	Memories	Comments
1100	MOV AX, 1212	More 1st number to AX register
1103	MOV BX, 1313	Move the 2nd number to BX register
1106	ADD AX, BX	Add both numbers
1108	MOV [1200], AX	Store result to memory location.



Input :

Memory location	Input Data
AX	1212
BX	1313

Output:

Memory location	Output Data
1200	25
1201	25

Subtraction of two numbers:

Algorithm :

Step 1 : Start the program.

step 2 : Load the 1st number to AX register

step 3 : Load the 2nd number to BX register

Step 4 : substract both the numbers.

Step 5 : Store the result in memory location

Step 6 : Halt the program.

Exp.No : 10

Date:

Simple Arithmetic Operations Using 8086 Microprocessors

Objective:

To develop 8086 assembly language programs for fundamental arithmetic operations, including addition, subtraction, multiplication and division and understand to manage operand sizes and handle overflow and carry conditions.

System and software tools required:

1. 8086 microprocessor kit
2. keyboard

Addition of two numbers:

Algorithm:

Step - 1 : Start the program.

Step - 2 : Load the 1st number to Ax register

Step - 3 : Load the 2nd number to Bx register

Step - 4 : Add both the numbers.

Step - 5 : Store the result in memory location

Step - 6 : End the program.