****

**Department of Electronics and Communication Engineering**

19EC408- MICROPROCESSOR AND MICROCONTROLLER LAB

**Index: List of Experiments**

**CYCLE I**

|  |  |
| --- | --- |
| S.NO. | TITLE OF THE EXPERIMENT |
|  | ARITHMETIC OPERATIONS USING 8086 MASM |
|  | INTERFACING STEPPER MOTOR WITH 8086 KIT AND ROTATE IT IN CLOCKWISE AND ANTI-CLOCKWISE DIRECTION. |
|  | INTERFACING DAC WITH 8086 KIT AND GENERATING TRIANGULAR AND SQUARE WAVEFORMS. |

**CYCLE II**

|  |  |
| --- | --- |
|  | FACTORIAL OF A NUMBER USING 8051 KEIL. |
|  | SORTING OF DATA - ASCENDING /DESCENDING ORDER USING 8051 KEIL. |
|  | SQUARE / CUBE OF 8 BIT DATA USING 8051 KEIL. |
|  | SERIAL TRANSFER OF SINGLE BYTE / CHARACTER USING 8051 KEIL.(EMBEDDED C PROGRAM) |
|  | TOGGLE AN LED ON/OFF BY USING AN EXTERNAL INTERRUPT IN ARM CONTROLLER |

**TABLE OF CONTENT**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| S.NO | DATE | TITLE OF THE EXPT | PAGE NO | MARK | SIGNATURE |
| 1 |  | ARITHMETIC OPERATIONS USING 8086 MASM |  |  |  |
| 2 |  | INTERFACING STEPPER MOTOR WITH 8086 KIT AND ROTATE IT IN CLOCKWISE AND ANTI-CLOCKWISE DIRECTION. |  |  |  |
| 3 |  | INTERFACING DAC WITH 8086 KIT AND GENERATING TRIANGULAR AND SQUARE WAVEFORMS. |  |  |  |
| 4 |  | FACTORIAL OF A NUMBER USING 8051 KEIL. |  |  |  |
| 5 |  | SORTING OF DATA - ASCENDING /DESCENDING ORDER USING 8051 KEIL. |  |  |  |
| 6 |  | SQUARE / CUBE OF 8 BIT DATA USING 8051 KEIL. |  |  |  |
| 7 |  | SERIAL TRANSFER OF SINGLE BYTE / CHARACTER USING 8051 KEIL.(EMBEDDED C PROGRAM) |  |  |  |
| 8 |  | TOGGLE AN LED ON/OFF BY USING AN EXTERNAL INTERRUPT IN ARM CONTROLLER. |  |  |  |

## ARITHMETIC OPERATIONS USING 8086 MICROPROCESSORS

##### EX NO: 1 DATE:

**AIM:** To write and execute Assembly language Program to perform arithmetic operations for 8086 microprocessor.

**APPARATUS REQUIRED:** Personal computer with MASM software

**1a)ASSEMBLY LANGUAGE PROGRAM BY DIRECT METHOD FOR ADDITION**

**ALGORITHM:**

**Step 1:** Open command prompt. **Step 2:** Type c: then cd masm › edit . **Step 3:** Then type the program.

**Step 4:** Initialize the memory location of 1st number.

**Step 5 :** Then increment the content of HL register pair and get second data

**Step 6:** Then perform the operation with accumulator and store the result in memory location.

**Step 7:** For output,type masm filename.asm,,;

link filename,,; debug filename.exe

**Step 8 :** Stop.

##### PROGRAM:

CODE SEGMENT

ASSUME CS: CODE,DS:CODE ORG 1000H

MOV CL,00H MOV AX,1234H MOV BX,1234H ADD AX,BX JNC L1

INC CL

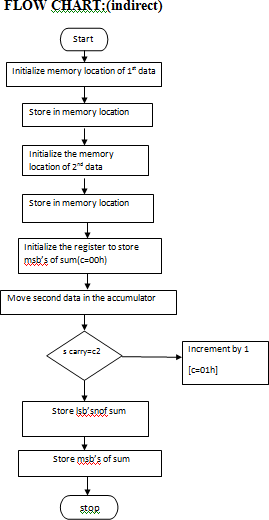
L1: MOV SI,1200H MOV [SI],AX MOV [SI+2],CL MOV AH,4CH INT 21H

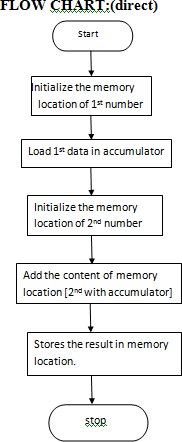
CODE ENDS END

**OUTPUT FOR ADDITION (DIRECT METHOD):**

|  |  |
| --- | --- |
| MEMORY LOCATION(INPUT) | MEMORY LOCATION(OUTPUT) |
|  |  |
|  |  |
|  |  |
|  |  |

##### Manual calculations:





**1b)ASSEMBLY LANGUAGE PROGRAM BY INDIRECT METHOD FOR ADDITION**

#### ALGORITHM:

**Step 1 :** Initialize the memory location for 1st data in HL register.

**Step 2 :** Store 1st data in memory location.

**Step 3 :** Increment the content of HL register for entering data in memory location.

**Step 4 :** Store second data in memory location.

**Step 5 :** Move 2nd number in accumulator.

**Step 6 :** Decrease the content of HL register pair.

**Step 7 :** Add the content of memory with accumulator.

**Step 8 :** Store the result in memory location.

**Step 9 :** Stop.

##### ASSEMBLY LANGUAGE PROGRAM :

CODE SEGMENT

ASSUME CS: CODE, DS:CODE ORG 1000H

MOV SI,2000H MOV CL,00H MOV AX,[SI] MOV BX,[SI+02H] ADD AX,BX

JNC ***L1***

INC CL

***L1***: MOV [SI+04H],AX MOV [SI+06H],CL MOV AH,4CH

INT 21H CODE ENDS END

**OUTPUT FOR ADDITION (INDIRECT METHOD):**

|  |  |
| --- | --- |
| MEMORY LOCATION(INPUT) | MEMORY LOCATION(OUTPUT) |
|  |  |
|  |  |
|  |  |
|  |  |

##### Manual calculations:

**2a)ASSEMBLY LANGUAGE PROGRAM BY DIRECT METHOD FOR SUBTRACTION**

#### ALGORITHM:

**Step 1 :** Load address of first number in MC register pair.

**Step 2 :** Move first data into accumulator.

**Step 3 :** Increment the content of MC register pair. **Step 4 :** Subtract the second data from accumulator. **Step 5 :** Store the result in Memory location.

##### PROGRAM :

CODE SEGMENT

ASSUME CS: CODE,DS:CODE ORG 1000H

MOV AX,1234H MOV BX,1234H SUB AX,BX

JNC **Down**

INC CL

**Down** : MOV SI,1200H

MOV [SI],AX MOV [SI+2],CL MOV AH,4CH INT 21H

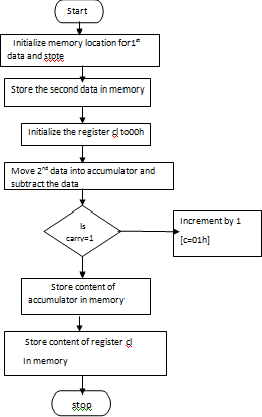
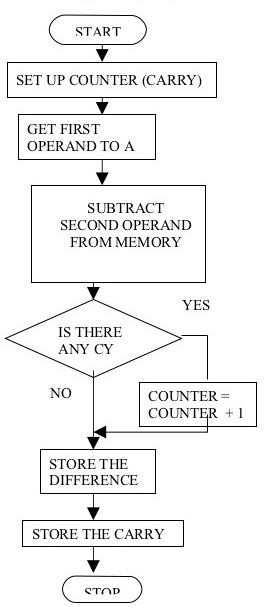
CODE ENDS END

**OUTPUT FOR SUBTRACTION**

|  |  |
| --- | --- |
| MEMORY LOCATION(INPUT) | MEMORY LOCATION(OUTPUT) |
|  |  |
|  |  |
|  |  |
|  |  |

##### Manual calculations:

**FLOW CHART (DITECT METHOD) FLOWCHART(INDIRECT METHOD)**



**2b) ASSEMBLY LANGUAGE PROGRAM BY INDIRECT METHOD**

#### ALGORITHM:

**Step 1 :** Initialize the memory location and store the 1st data.

**Step 2 :** Increment the content of AL and get the 2nd data.

**Step 3 :** Move the 2nd data in accumulator.

**Step 4 :** Subtract the content of memory and store the result.

##### PROGRAM :

CODE SEGMENT

ASSUME CS: CODE,DS:CODE ORG 1000H

MOV SI,2000H MOV CL,00H MOV AX,[SI] MOV BX,[SI+02H] SUB AX, BX

JNC ***DOWN***

INC CL

***DOWN:*** MOV [SI+04H],AX

MOV [SI+06H],CL MOV AH,4CH

INT 21H CODE ENDS

END

**OUTPUT FOR SUBTRACTION (Indirect Method)**

|  |  |
| --- | --- |
| MEMORY LOCATION(INPUT) | MEMORY LOCATION(OUTPUT) |
|  |  |
|  |  |
|  |  |
|  |  |

**Manual calculations:**

**3a) ASSEMBLY LANGUAGE PROGRAM BY DIRECT METHOD FOR MULTIPLICATION**

#### ALGORITHM:

**Step 1 :** Move the first number into the accumulator.

**Step 2 :** Increment the content of register pair.

**Step 3 :** Multiply both the data.

**Step 4 :** Store the result in memory location.

##### PROGRAM :

CODE SEGMENT

ASSUME CS: CODE,DS:CODE ORG 1000H

MOV DX,0000H MOV AX,1234H MOV BX,1234H MUL BX

MOV SI,1200H MOV [SI],AX MOV [SI+02H],DX MOV AH,4CH

INT 21H CODE ENDS END

**OUTPUT FOR MULTIPLICATION(DIRECT METHOD)**

|  |  |
| --- | --- |
| MEMORY LOCATION(INPUT) | MEMORY LOCATION(OUTPUT) |
|  |  |
|  |  |
|  |  |
|  |  |

##### Manual calculations:

**3b)ASSEMBLY LANGUAGE PROGRAM BY INDIRECT METHOD FOR MULTIPLICATION**

#### ALGORITHM:

**Step 1 :** initialize the memory location of

**Step 2 :** increment the context of and get 2nd data

**Step 3 :** move the 2nd data in a accumulator.

**Step 4 :** multiply the contents & store the result.

##### PROGRAM :

CODE SEGMENT

ASSUME CS: CODE,DS:CODE

ORG 1000H

MOV SI,2000H

MOV DX,0000H

MOV AX,[SI]

MOV BX,[SI+02H]

MUL BX

MOV [SI+04H],AX

MOV [SI+06H],DX

MOV AH,4CH

INT 21H

CODE ENDS

END

**OUTPUT FOR MULTIPLICATION (INDIRECT METHOD)**

|  |  |
| --- | --- |
| MEMORY LOCATION(INPUT) | MEMORY LOCATION(OUTPUT) |
|  |  |
|  |  |
|  |  |
|  |  |

##### Manual calculations:

**4a)ASSEMBLY LANGUAGE PROGRAM BY DIRECT METHOD FOR DIVISION**

#### ALGORITHM:

**Step 1 :** load first no in register ax

**Step 2 :** store the 2nd data In register bx.

**Step 3 :** divide the content of ax by cx and result in ax.

##### PROGRAM :

CODE SEGMENT

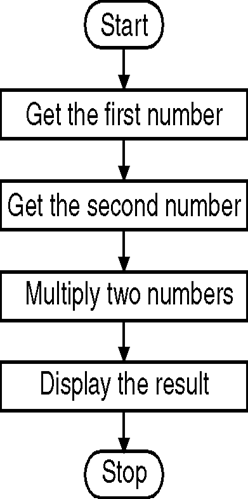
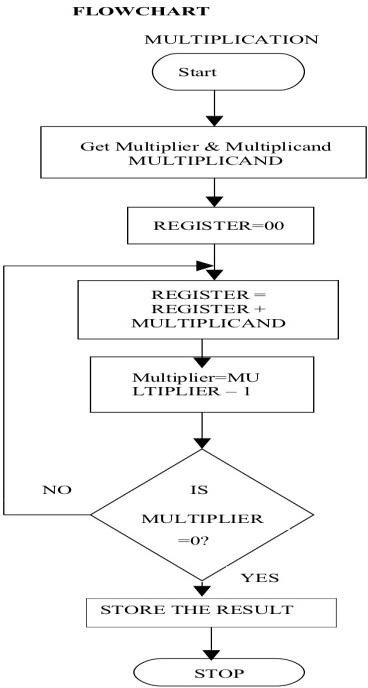
ASSUME CS: CODE,DS:CODE ORG 1000H

MOV DX,0000H MOV AX,1234H MOV BX,1234H DIV BX

MOV SI,1200H MOV [SI],AX MOV [SI+02H],DX MOV AH,4CH

INT 21H CODE ENDS END

**Flowchart(INDIRECT METHOD) Flow chart (DIRECT METHOD)**



##### OUTPUT FOR DIVISION(DIRECT METHOD)

|  |  |
| --- | --- |
| MEMORY LOCATION(INPUT) | MEMORY LOCATION(OUTPUT) |
|  |  |
|  |  |
|  |  |
|  |  |

**Manual calculations:**

##### 4b)ASSEMBLY LANGUAGE PROGRAM BY INDIRECT METHOD FOR DIVISION

**ALGORITHM:**

**Step 1 :** load memory location of data in the reg pair **Step 2 :** move data to accumulator & perform division **Step 3 :** store the result.

##### PROGRAM:

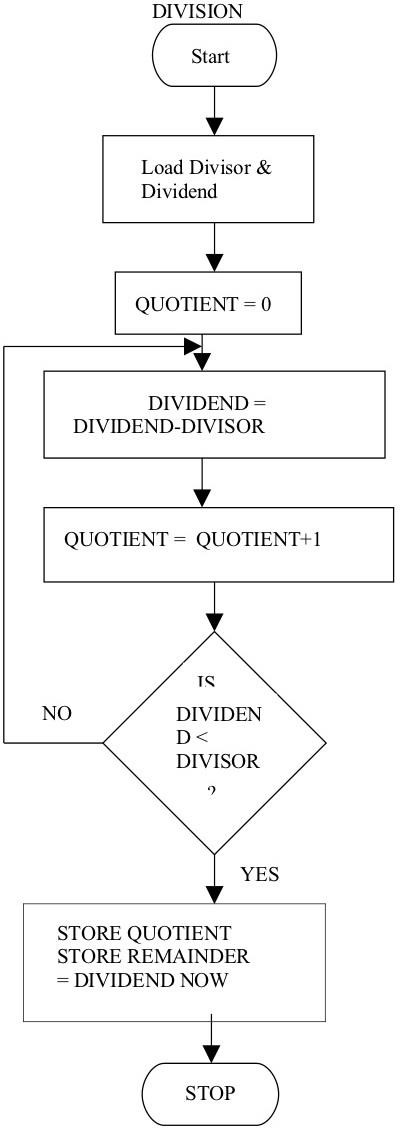
CODE SEGMENT

ASSUME CS: CODE,DS:CODE ORG 1000H

MOV SI,2000H MOV DX,0000H MOV AX,[SI] MOV BX,[SI+02H] DIV BX

MOV [SI+04H],AX MOV [SI+06H],DX MOV AH,4CH

INT 21H CODE ENDS END

**Flowchart (DIRECT METHOD): Flowchart (INDIRECT METHOD):**

STOP

STORE THE RESULT IN ACCUMULATOR

DIVIDE THE CONTENT OF AX BY BX USING DIV INSTRUCTION

LOAD THE DIVISOR IN REGISTER BX

LOAD THE DIVIDEND IN REGISTER AX

START

**REG NO:**

##### OUTPUT FOR DIVISION(INDIRECT METHOD)

|  |  |
| --- | --- |
| MEMORY LOCATION(INPUT) | MEMORY LOCATION(OUTPUT) |
|  |  |
|  |  |
|  |  |
|  |  |

**Manual calculations:**

##### RESULT :

Thus the Assembly language Programs of 8086 to perform arithmetic operations in both direct and indirect methods was written and executed in MASM software.

## STEPPER MOTOR INTERFACING

##### EX.NO:2 DATE:

**AIM:** To write an assembly language program in 8086 to rotate the motor at different speeds.

##### APPARATUS REQUIRED:

|  |  |  |  |
| --- | --- | --- | --- |
| SL.NO | **ITEM** | SPECIFICATION | QUANTITY |
| 1. | **Microprocessor kit** | 8086 | 1 |
| 2. | **Power Supply** | +5 V, dc,+12 V dc | 1 |
| 3. | **Stepper Motor Interface board** | - | 1 |
| 4. | **Stepper Motor** | - | 1 |

##### THEORY:

A motor in which the rotor is able to assume only discrete stationary angular position is a stepper motor. The rotary motion occurs in a stepwise manner from one equilibrium position to the next. Two-phase scheme: Any two adjacent stator windings are energized. There are two magnetic fields active in quadrature and none of the rotor pole faces can be in direct alignment with the stator poles. A partial but symmetric alignment of the rotor poles is of course possible.

**ALGORITHM:**

For running stepper motor clockwise and anticlockwise directions

**Step 1:** Get the first data from the lookup table.

**Step 2:** Initialize the counter and move data into accumulator.

**Step 3:** Drive the stepper motor circuitry and introduce delay

**Step 4:** Decrement the counter is not zero repeat from step(iii)

**Step 5:** Repeat the above procedure both for backward and forward directions.

##### SWITCHING SEQUENCE OF STEPPER MOTOR:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| MEMORY LOCATION | A1 | A2 | B1 | B2 | HEX CODE |
| 4500 | 1 | 0 | 0 | 0 | 09 H |
| 4501 | 0 | 1 | 0 | 1 | 05 H |
| 4502 | 0 | 1 | 1 | 0 | 06 H |
| 4503 | 1 | 0 | 1 | 0 | 0A H |

**PROGRAM TABLE**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| MEMORY LOCATION | OPCODE |  | PROGRAM | COMMENTS |
| 1000 | C7 C7 00 12 | START : | MOV DI, 1200H | Initialize memory location to store the array of number |
| 1004 | C7 C1 04  00 |  | MOV CX, 0004H | Initialize array size |
| 1008 | 8A 05 | DOWN 1 : | MOV AL,[DI] | Copy the first data in AL |
| 100A | E6 C0 |  | OUT C0,AL | Send it through port address |
| 100C | C7 C2 10  10 |  | MOV DX, 1010H | Introduce delay |
| 1010 | 4A | L1 : | DEC DX |
| 1011 | 75 FD |  | JNZ L1 |
| 1013 | 47 |  | INC DI | Go to next memory location |
| 1014 | E2 F2 |  | LOOP DOWN1 | Down until all the data’s have been sent |
| 1016 | E9 E7 FF |  | JMP START | Go to start location for continuous rotation |
| 1019 | F4 |  | HLT | stop |
| 1200 | 09,05,06,0A |  |  | Array of data |

**RESULT:** Thus the assembly language program for rotating stepper motor in both clockwise

and anticlockwise directions was written and verified.

## INTERFACING DAC WITH 8086 KIT AND GENERATING SAWTOOTH AND SQUARE WAVEFORMS

##### EX.NO:3 DATE:

**AIM:** To write an assembly language program in 8086 to generate Sawtooth and square waveforms using DAC.

##### APPARATUS REQUIRED:

|  |  |  |  |
| --- | --- | --- | --- |
| SL.NO | **ITEM** | SPECIFICATION | QUANTITY |
| 1. | Microprocessor kit | 8086 | 1 |
| 2. | Power Supply | +5 V dc,+12 V dc | 1 |
| 3. | DAC Interface board | - | 1 |

**ALGORITHM:**

**Measurement of analog voltage:**

**Step I :** Send the digital value of DAC.

**Step II :** Read the corresponding analog value of its output.

**Waveform generation:**

Square Waveform:

**Step I :** Send low value (00) to the DAC.

**Step II :** Introduce suitable delay.

**Step III :** Send high value to DAC.

**Step IV :** Introduce delay.

**Step V :** Repeat the above procedure.

Saw-tooth waveform:

**Step I :** Load low value (00) to accumulator.

**Step II :** Send this value to DAC.

**Step III :** Increment the accumulator.

**Step IV :** Repeat step (ii) and (iii) until accumulator value reaches FF.

**Step V :** Repeat the above procedure from step 1.

**Program: Square wave**

|  |  |  |  |
| --- | --- | --- | --- |
| **MEMORY LOCATION** | **OPCODE** | **PROGRAM** | **COMMENTS** |
| 1000 | C6 C0 36 | MOV AL,36H | Load 36 in Accumulator |
| 1003 | E6 CC | OUT 0CCH,AL | Send through output port |
| 1005 | C6 00 0A | MOV AL,10H | Load 10 in Accumulator |
| 1008 | E6 C8 | OUT 0C8H,AL | Send through output port |
| 100A | C6 C0 00 | MOV AL,00H | Load count value in AL register |
| 100D | E6 C8 | OUT 0C8H,AL | Send through output port |
| 100F | F4 | HLT | Stop |

**Program: Sawtooth wave**

|  |  |  |  |
| --- | --- | --- | --- |
| MEMORY  LOCATION | OPCODE | PROGRAM | COMMENTS |
| 1000 | C6 C0 00 | START: MOV AL,00H | Load 00 in accumulator |
| 1003 | E6 C8 | LOOP : OUT 0C8H,AL | Send through output port |
| 1005 | FE C0 | INC AL | Increment contents of accumulator |
| 1007 | 73 FA | JNC LOOP | Send through output port |
| 1009 | E9 F4 FF | JMP START | Go to starting location |

**Tabulation:**

|  |  |  |
| --- | --- | --- |
| Waveform | Amplitude | Time period |
| Square |  |  |
| Sawtooth |  |  |

**Model Graph:**

**Result:**

Thus the DAC was interfaced with 8086 and different waveforms have been generated.

## FACTORIAL OF A NUMBER USING 8051 KEIL

##### EX NO: 4 DATE:

**AIM:** To write and execute Assembly language Program to perform factorial of a number using 8051 keil.

##### APPARATUS REQUIRED: Personal computer with Keil software

**ALGORITHM:**

 **Start**

 **Input**: Read the number n.

 **Initialize**:

* Set factorial to 1.
* Set i to 1.

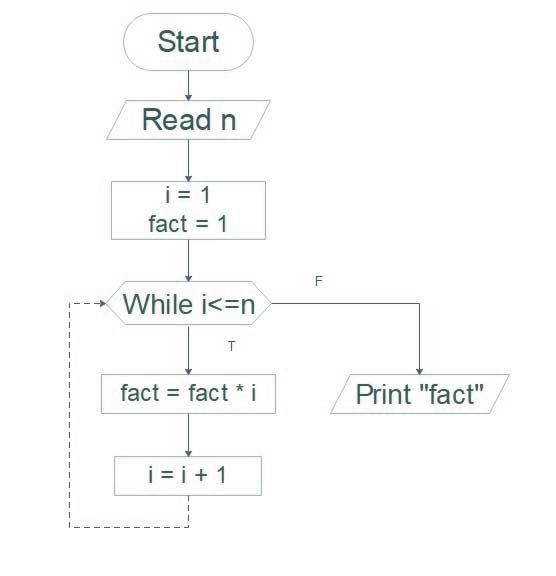
 **Loop**: While i is less than or equal to n:

* Multiply factorial by i.
* Increment i by 1.

 **Output**: Store or print the value of factorial.

 **End**

**FLOW CHART:**



**Program:**

ORG 0000H

MOV DPTR,#4500H

MOVX A,@DPTR

INC DPTR

MOV R0,A

ACALL FACTORIAL

MOVX @DPTR,A

SJMP THIN

FACTORIAL:DEC R0

CJNE R0,#01H,PRODUCT

SJMP THICK

PRODUCT:MOV B,R0

MUL AB

ACALL FACTORIAL

THICK: RET

THIN:RET

END

**Output:**

**Manual Calculations:**

**Result:**

Thus the factorial of a number using 8051 keil was calculated and shown the output.

**SORTING OF DATA - ASCENDING /DESCENDING ORDER USING 8051 KEIL**

##### EX NO: 5 DATE:

**AIM:** To write and execute Assembly language Program for sorting of data using 8051 keil.

##### APPARATUS REQUIRED: Personal computer with Keil software

**(i) Descending order**

**ALGORITHM:**

1. Initialize the register r7 with count.

2. Get first two elements in two registers.

3. Compare the two elements of data. If value of R0 register is low, then exchange A

& R0 data else increment pointer and decrement register R7.

4. Check R7 is zero, and then move the register R0 & A.

5. Again increment pointer and decrement R7,

6. Check R7 is zero. If no repeat the process from step 2.

7. Otherwise stop the program.

**PROGRAM:**

ORG 0000H

LOOP1:MOV R0,#40H

MOV R6,#06H

DEC R6

LOOP:MOV A,@R0

INC R0

MOV B,@R0

CJNE A,B,NEXT

NEXT:JC DOWN

MOV@R0,A

DEC R0

MOV@R0,B

INC R0

DOWN:DJNZ R6,LOOP

MOV R1,#02H

DJNZ R1,LOOP1

END

**OUTPUT:**

MEMORY WINDOW:

Before execution:

After execution:

**(i) Ascending order**

**ALGORITHM:**

1. Initialize the register r7 with count.

2. Get first two elements in two registers.

3. Compare the two elements of data. If value of R0 register is high then exchange A

& R0 data else increment pointer and decrement register R7.

4. Check R7 is zero, and then move the register R0 & A.

5. Again increment pointer and decrement R7,

6. Check R7 is zero. If no repeat the process from step 2.

7. Otherwise stop the program.

**PROGRAM:**

ORG 0000H

LOOP1:MOV R0,#40H

MOV R6,#06H

DEC R6

LOOP:MOV A,@R0

INC R0

MOV B,@R0

CJNE A,B,NEXT

NEXT:JNC DOWN

MOV@R0,A

DEC R0

MOV@R0,B

INC R0

DOWN:DJNZ R6,LOOP

MOV R1,#02H

DJNZ R1,LOOP1

END

**OUTPUT:**

MEMORY WINDOW:

Before execution:

After execution:

**Result:**

Thus the sorting of given data was done using 8051 keil and shown the output.

##### SQUARE OF 8 BIT DATA USING 8051 KEIL

##### EX.NO:6 DATE:

**AIM:** To write and execute Assembly language Program for finding the square of given data using

8051 keil

##### APPARATUS REQUIRED: Personal computer with Keil software

**ALGORITHM:**

1. Enter a program.

2. Enter the input value to P0

3. Execute the program.

4. The output square value is stored in P2

**PROGRAM:**

MOV A,P0

MOV R0,A

MOV B,R0

MUL AB

MOV P2,A

END

**OUTPUT:**

**Result:**

Thus the square of given data was done using 8051 keil and shown the output.

##### SERIAL TRANSFER OF SINGLE BYTE / CHARACTER USING 8051

##### KEIL.(EMBEDDED C PROGRAM)

##### EX.NO:7 DATE:

##### AIM: To write and execute Embedded C Program for Serial Transfer of Single Byte / Character using 8051 KEIL

##### APPARATUS REQUIRED: Personal computer with Keil software

**PROGRAM:**

**(i)Serial port transfer a character A**

**Using C Program**

#include<reg51.h>

void main(void)

{

TMOD=0X20;//TIMER 1,MODE 2

TH1=0XFA;

SCON=0X50;

TR1=1;

while(1)

{

SBUF='A';

while(TI==0);

TI=0;

}

}

**Using 8051 Instructions Program**

ORG 00H

MOV TMOD, #20H

MOV TH1, #0FDH

MOV SCON, #50H

SETB TR1

MAIN\_LOOP:

MOV SBUF, #'B'

WAIT\_TI:

JNB TI, WAIT\_TI

CLR TI

SJMP MAIN\_LOOP

END

**(ii) Serial port to Transfer a Message**

**Using C Program**

#include<reg51.h>

void main(void)

{

unsigned char msg[]="Programming 8051";

unsigned char i;

TMOD=0X20;//TIMER 1,MODE 2

TH1=0XFA;

SCON=0X50;

TR1=1;

for (i=0; i<17;i++)

{

SBUF= msg[i];

while(TI==0);

TI=0;

}

while(1);

}

**Using 8051 Instructions Program**

ORG 00H

START:

MOV TMOD, #20H

MOV TH1, #0FAH

MOV SCON, #50H

SETB TR1

MOV DPTR, #4500H

SEND\_MESSAGE: MOVX A, @DPTR

JZ END\_TRANSMIT

MOV SBUF, A

WAIT\_TI:

JNB TI, WAIT\_TI

CLR TI

INC DPTR

SJMP SEND\_MESSAGE

END\_TRANSMIT:

SJMP END\_TRANSMIT

L1:SJMP L1

**OUTPUT:**

**Result:**

##### Thus the Serial transfer of Single Byte / Character using 8051 KEIL was done and shown the output.

##### TOGGLE AN LED ON/OFF BY USING AN EXTERNAL INTERRUPT

##### IN ARM CONTROLLER

##### EX.NO:8 DATE:

**Aim: To Interface a Digital Input (user push button ) to ARM development board and write a program to obtain the data and flash the led.**

##### Components required:

1. **STM32 CUBE IDE**

##### ARM IOT development board

1. **STM programmer tool. Theory**

The full form of an ARM is an advanced reduced instruction set computer (RISC) machine, and it is a 32-bit processor architecture expanded by ARM holdings. The applications of an ARM processor include several microcontrollers as well as processors. The architecture of an ARM processor was licensed by many corporations for designing ARM processor-based SoC products and CPUs. This allows the corporations to manufacture their products using ARM architecture. Likewise, all main semiconductor companies will make ARM-based SOCs such as Samsung, Atmel, TI etc.

##### Procedure:

1. Click on STM 32 CUBE IDE.
2. Click on FILE, click on new stm 32 project.
3. Select the target to be programmed as shown below and click on next.
4. Select the program name and save it.
5. Corresponding ioc file will be generated automatically.
6. Select the appropriate pins as gipo, in or out, USART or required options and configure.
7. click on cntrl+S , automaticall C program will be generated.
8. Edit the program and as per required.
9. Use project and build all.
10. Once the project is build link the hexfile build in stm32cube ide using post processor build.
11. Click on debug option.
12. Connect the stm nucleo board and click on run.

##### STM 32 CUBE PROGRAM

#include "main.h" #include "stdbool.h" bool buttonstatus; void pushbutton();

void SystemClock\_Config(void); static void MX\_GPIO\_Init(void); int main(void)

{

HAL\_Init(); SystemClock\_Config(); MX\_GPIO\_Init(); while (1)

{

pushbutton();

}

}

void pushbutton()

{

buttonstatus=HAL\_GPIO\_ReadPin(GPIOC, GPIO\_PIN\_13); if(buttonstatus==0)

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

HAL\_Delay(200);

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, GPIO\_PIN\_SET);

HAL\_Delay(200);

}

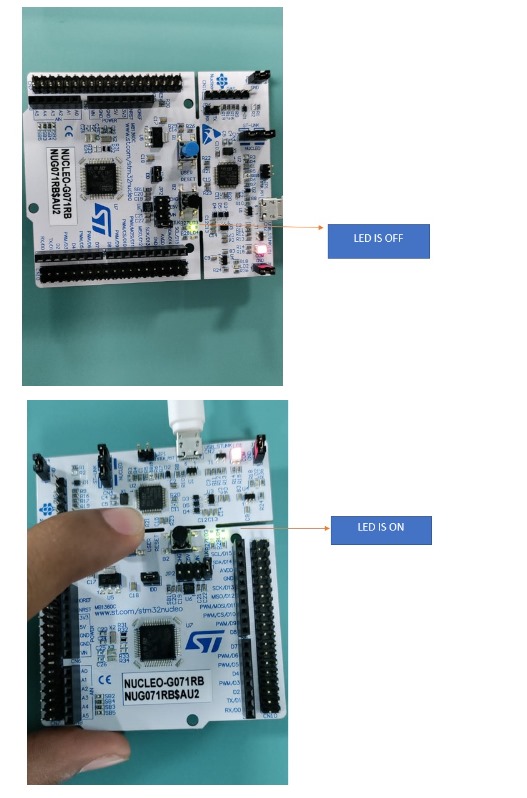
else

{

HAL\_GPIO\_WritePin(GPIOA, GPIO\_PIN\_5, GPIO\_PIN\_RESET);

}

} **OUTPUT:**



**Result :**

Interfacing a digital Input (Pushbutton ) with ARM microcontroller based IOT development is executed and the results are verified.