K. L. E. SOCIETY'S

K.L.E. INSTITUTE OF TECHNOLOGY,

GOKUL, HUBBALLI-580027.

CONTENTS

Sl. No.	NAME OF THE EXPERIMENT	Page No.	Date	Remarks
	Introduction	1–6		
1.	Write an ALP to move a block of n bytes	7–8	25/2/25	Madhu
	of data from source (20H) to destination (40h)			
	using internal RAM.			
2.	Write an ALP to move a block of n-bytes	9–11	25/2/25	
	of data from source (2000h) to destination			
	(2050h) using external RAM.			
3.	Write an ALP to exchange the source block	12–13	4/3/25	Madhu
	containing n bytes of data with destination			
	starting with address 40h.			
4.	Write an ALP to add byte in the RAM at	14–15	4/3/25	
	34h & 35h, store the result in the register			
	R5 and R6 using indirect add.			
5.	Write an ALP to exchange the source block	16–17	25/3/25	Madhu
	containing N bytes of data with destination			
	starting with location 00h.			
6.	Write an ALP to subtract the bytes in	18–19	25/3/25	
	internal RAM 34h & 31h store the result in			
	register R5 (LSB) & R6 (MSB)			
7.	Write an ALP to multiply two 8bit nos	20–21	1/4/25	Madhu
	stored at 30h & 31h and store 16 bit result			
	in 32h & 33h of internal RAM.			
8.	Write an ALP to perform division operation	22–23	1/4/25	
	on 8-bit number by 8bit number.			

Madhu

STAFF-IN-CHARGE H.O.D. PRINCIPAL

K. L. E. SOCIETY'S

K.L.E. INSTITUTE OF TECHNOLOGY,

GOKUL, HUBBALLI-580027.

CONTENTS

Sl. No.	NAME OF THE EXPERIMENT	Page No.	Date	Remarks
9.	Write an ALP to seperate Positive & nigative in a given array.	24-25	29/4/25	Madhu
10.	Write an ALP to seperate odd and even elements in a given array.	26-27	29/4/25	
11.	Write an ALP to arrange the numbers in ascending and descending order.	28-30	6/5/25	Madhu
12.	Write an ALP to find largest and smallest no. from a given array storing from 20h and store in internal memory 40h.	31-33	6/5/25	
13.	Write an ALP for decimal up counter.	34-35	13/5/25	Madhu
14.	Write an ALP for decimal down counter.	36-37	13/5/25	
15.	Write an ALP for hexadecimal up counter.	38-39	13/5/25	
16.	Write an ALP for hexadecimal down counter.	40-41	13/5/25	
17.	Write an 8051 c Program to find the sum of first 10 integers.	42-43	20/5/25	Madhu
18.	Write an 8051 c Program to find the factorial of a given number.	44-45	20/5/25	
19.	Write an 8051 c Program to find the square of a no. using look up table.	46-47	20/5/25	
20	Write an 8051 c Program to count no. of one's and zeroes in 2 consecutive mem location.	48-50	20/5/25	
21.	Write an 8051 C Program to rotate stepper motor.	51-53	20/5/25	Madhu
22.	Write an 8051 c Prog to generate sine and square waveforms using DAC interface	54-57	20/5/25	

Madhu

STAFF-IN-CHARGE H.O.D. PRINCIPAL

Introduction to Keil µvision version 5:

Keil µvision is a popular integrated development environment (IDE) for embedded system development. Specifically designed for microcontrollers, version 5 is a significant update that offers several new features and improvement.

* features of Keilvision version 5:

1. Project management:
   A new project explorer that simplifies Project creation management and debugging.

2. code editor:
   An improved code editor with features like syntax highlighting code completion & code refactoring.

3. Debugger:-
   A powerful debugger that allows develops to debug their code in real-time.

4. compiler:-
   A high-Performance compiler that optimizes code for size speed and power consumption.

5. Simulator:-
   A simulator that allows developers to test and debug their code without the need for physical hardware.

* Benefits of Keil µVision version 5

1. Improved Productivity:-
   Enhanced user interface & debugging tools improve developer Productivity & efficiency.

2. Increases Performance:-
   New compiler optimizations and simulator improve code performance & reduce size.

3. Enhanced debugging:-
   Advanced debugging features & real-time data visualization improve debugging and troubleshooting.

4. Support for latest devices:-
   Support for new microcontrollers and devices ensures that developers can create applications for the latest hardware.

* System requirements:-

1. Operating system: windows 7+
2. Processor - Intel core i3 or higher
3. Memory: 4GB RAM or higher
4. Disk space: 2GB free disk space or higher.

K.L.E. INSTITUTE OF TECHNOLOGY, HUBBALLI.
Page No.:
Date :

* Applications:-

1. Embedded systems:- Keil µvision is widely used in the development of embedded system, including microcontrollers

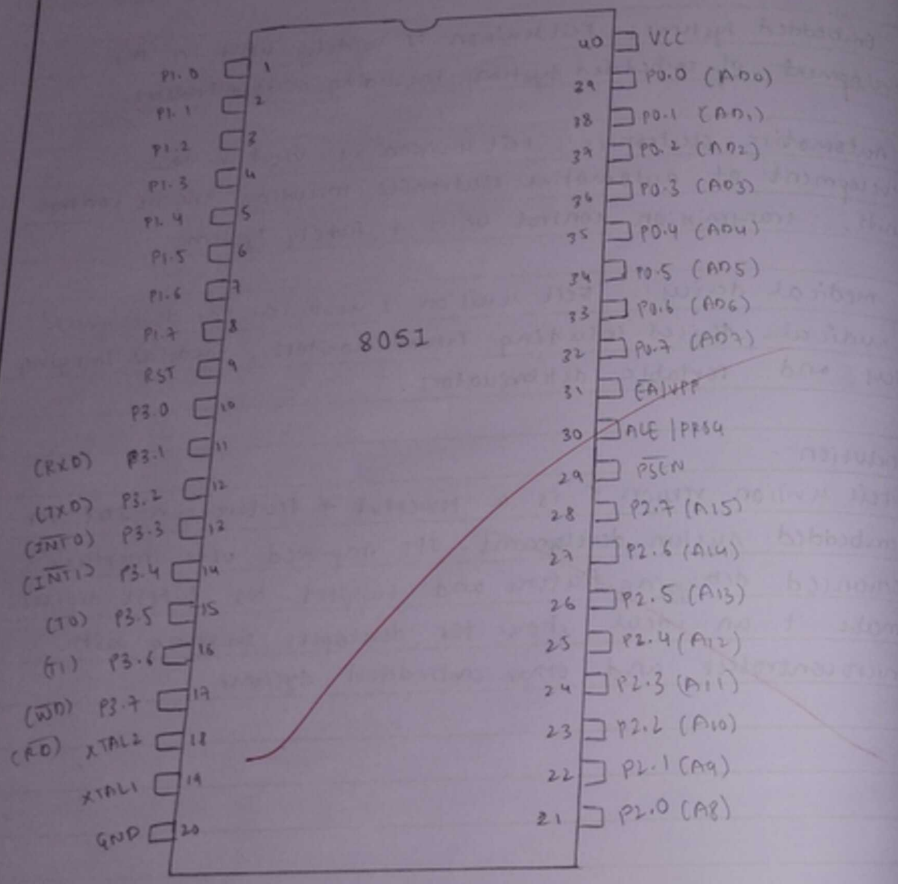
2. Automotive electronics:- Keil µvision is used in the development of automotive electronics including engine control units, transmission control units & safety system.

3. Medical devices:- Keil µvision is used in the development of medical devices including Patient monitors, medical imaging devices and portable defibrillators.

conclusion:-

Keil µvision version 5 is a powerful & feature rich IDE for embedded system development. Its improved user interface, advanced debugging features and support for latest devices make it an ideal choice for developers working with microcontroller and other embedded systems.

Pin diagram of 8051 microcontroller



Introduction to 8051 microcontroller.

The 8051 microcontroller is a popular and widely used microcontroller family developed by intel, it is smart computer on a single integrated circuit (IC) that contains a processor. memory and input/output peripherals. It is an 8-bit microcontroller that is known for its simplicity ease of use, and versatility, microcontroller are used in wide range of applications from single simple embedded system to complex robotics & IOT devices.

- Architecture of 8051 microcontroller.

The 8051 microcontroller has a harvard architecture which means that it has seperate buses for program and data memory, the architecture consist of.

1. CPU :- The central processing unit (CPU) is the brain of the microcontroller & executes instructions.
2. Program memory :- The program memory stores the program instructions & is typically implemented using flash memory or ROM.
3. Data memory :- the data memory stores data used by the program and is typically implemented using RAM.
4. Input/output (I/O) ports :- The I/O ports provide a way for the microcontrollers to interact with external devices.

features of 8051 microcontrollers :-

The 8051 microcontroller has several features that make it a popular choice for many applications.

(1). 8 bit Architecture :-
The 8051 microcontroller has an 8-bit architecture, which means that it can process 8 bit data.

(2). 16 bit Address Bus :-
the 8051 microcontroller has a 16-bit address bus, which allows it to address up to 64KB of memory.

(3). 8KB Program memory :-
The 8051 microcontroller has 8KB of Program memory, which can be expanded using external memory.

(4). 256 bytes Data memory :-
the 8051 microcontroller has 256 bytes of data memory which can be expanded using external memory.

(5). 32 I/O lines :-
the 8051 microcontroller has 32 I/O lines can be used to interact with external devices.

(6). Two 16-bit timers/counters :-
the 8051 microcontroller has two 16 bit timers/counters that can be used to generate timing signals for count external events.

(7). full-duplex serial interface :-
The 8051 microcontroller has full duplex serial interface that allows it to communicate with other devices.

Pin Diagram of 8051 micro controller.

- Pin 1-8 :- These pins are known as port 1, the port doesn't serve any other functions, it is internally pulled up, bi-directional I/o Port.
- Pin 9 :- It is a RESET Pin, which is used to reset the micro controller to its initial values.
- Pin 10-17 :- these pins are known as port 3, this port serves some functions like interrupt, timer input, control signals, serial communication signal RXD & TXD.
- Pin 18-19 :- These pins are used for interfacing an external crystal to get the system clock.
- pin 20 :- This pin provide power supply is the circuit.
- Pin 21-28 :- These pins are known as port 2, it serves as I/o port, higher order address bus signals are also multiplexed using the Port.
- Pin 29 :- This is PSEN pin which stands for Program store enable, it is used to read a signal from external Program memory.
- Pin 30 :- This is an pin which is stands for external access input, It is used to enable/disable the external memory interfacing.
- Pin 31 :- This is ALE which is stands for address latch enable. It is used to demultipler the address, data signal port.
- Pin 32-39 :- These Pins are known as port 0 i serves as I/o port lower Port Order address & data bus signals are multiplexed using the Port.
- Pin 40 :- This pin used to provide power supply to the circuits.

* Applications of 8051 microcontroller :-

(1). Embedded systems :-
The 8051 microcontroller is used in many embedded systems including robotics, automotive electronics & medical devices.

(2). Industrial control systems :-
The 8051 microcontroller is used in industrial control system including process control, motor control & power management.

(3). consumer electronics :-
The 8051 microcontroller is used in many consumer electronics including appliances, toy & games.

conclusion :-

The 8051 microcontroller is a popular and widely used microcontroller family that is known for its simplicity, ease of use and versatility its architecture & features make it a great choice for many applications including embedded system, industrial control systems & consumer electronics.

EXPERIMENT-01

Algorithm:-

Step 1:- Load a register with address of source 20h

Step 2:- Load another register with address of destination 40h

Step 3:- Load 3rd with number of block to moved n=5

Step 4:- Loop initialization is done i.e target is built Load accumulator with data pointer by R0 register

Step 5:- Load the current content to A to the current destination

Step 6:- Increment source address register by byte

Step 7:- Increment destination address by one byte.

Step 8:- Loop back & decrement the value of counter register jump to target if value is not equal to zero. The program execution reaches here where all block have been moved hence Program module ends.

Memory 1

d 0x20

D:0x20: 01 02 03 04 05 00 00
D:0x27: 00 00 00 00 00 00 00
D:0x2E: 00 00 00 00 00 00 00
D:0x35: 00 00 00 00 00 00 00
D:0x3C: 00 00 00 00 00 00 00

Memory 2

d 0x40

D:0x40: 00 00 00 00 00 00 00
D:0x47: 00 00 00 00 00 00 00
D:0x4E: 00 00 00 00 00 00 00
D:0x55: 00 00 00 00 00 00 00
D:0x5C: 00 00 00 00 00 00 00

Memory 1

d 0x20

D:0x20: 01 02 03 04 05 00 00
D:0x27: 00 00 00 00 00 00 00
D:0x2E: 00 00 00 00 00 00 00
D:0x35: 00 00 00 00 00 00 00
D:0x3C: 00 00 00 00 01 02 03
D:0x43: 04 05 00 00 00 00 00

Memory 2

d 0x40

D:0x40: 01 02 03 04 05 00 00
D:0x47: 00 00 00 00 00 00 00
D:0x4E: 00 00 00 00 00 00 00
D:0x55: 00 00 00 00 00 00 00
D:0x5C: 00 00 00 00 00 00 00
D:0x63: 00 00 00 00 00 00 00

Experiment - 01

Aim:- write an ALP to mov a block OR n bytes OR data from source (20H) to destination (40H) using internal RAM.

Resource Required :- Software Keil - µvision

Program:-

```
ORG 0000H
MOV R2, #05H
MOV R0, #20H
MOV R1, #40H
UP: MOV A, @R0
    MOV @R1, A
    INC R0
    INC R1
    DJNZ R2, UP
    . END
```

Observation:-

Before execution

mem	data	mem	data
20H	01	40H	00
21H	02	41H	00
22H	03	42H	00
23H	04	43H	00
24H	05	44H	00

After execution

mem	data	mem	data
20H	01	40H	01
21H	02	41H	02
22H	03	42H	03
23H	04	43H	04
24H	05	44H	05

Result and conclusion:-

moving block of 5 bytes from 20H to 40H using internal RAM has been verified using Keil µvision software

EXPERIMENT-02

Algorithm:-

Step 1:- Load source register in data pointer register.

Step 2:- Load destination address in 16-bit in register Point

Step 3:- Load number of data block to be moved in register.

Step 4:- Loop initialization to move single bit data from external memory accumulator

Step 5:- PUSH DPH & DPL into the stack.

Step 6:- move destination address to DPH & DPL

Step 7:- move the data from accumulator to address pointed external memory

Step 8:- Increment the data pointer by 1-byte

Step 9:- move DPH & DPL data to destination address

Step 10:- POP of the DPH & DPL data out of the stack.

Step 11:- Increment the data pointer value again.

Step 12:- Decrement & jump the value of counter register to target value is not equal to zero

Step 13:- This ends program module

Experiment-02

Aim:- write an ALP to move a block of n-bytes of data from source (2000h) to destination (2050h) using external RAM.

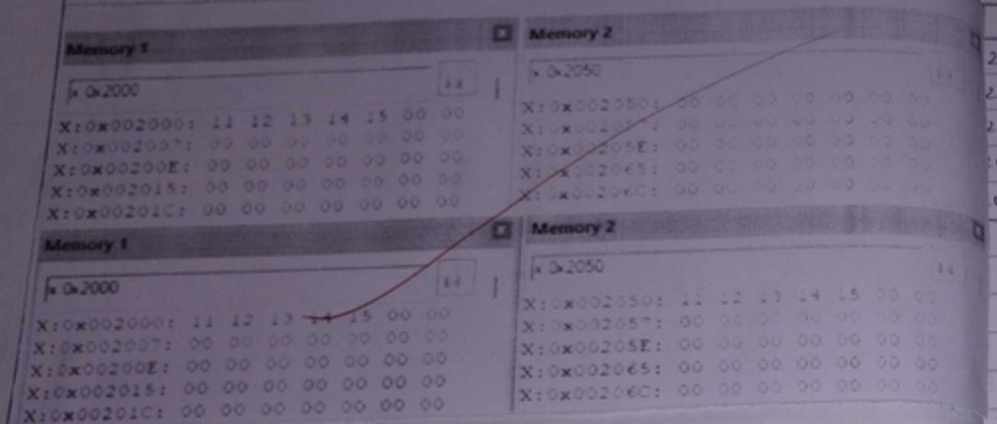
Resource required :- software - keil µvision

Program :-

```
MOV DPTR, #2000H
MOV R0, #20H
MOV R1, #50H
MOV R2, #05H
UP: MOVX A, @DPTR
PUSH DPH
PUSH DPL
MOV DPH, R0
MOV DPL, R1
MOVX @DPTR, A
INC DPTR
MOV R0, DPH
MOV R1, DPL
POP DPL
POP DPH
INC DPTR
DJNZ R2, UP
END.
```

Result and conclusion :-

The ALP effectively moves data from source to destination using external data.



Observation

Before execution

mem	data	mem	data
2000H	11	2050H	00
2001H	12	2051H	00
2002H	13	2052H	00
2003H	14	2053H	00
2004H	15	2054H	00

After execution

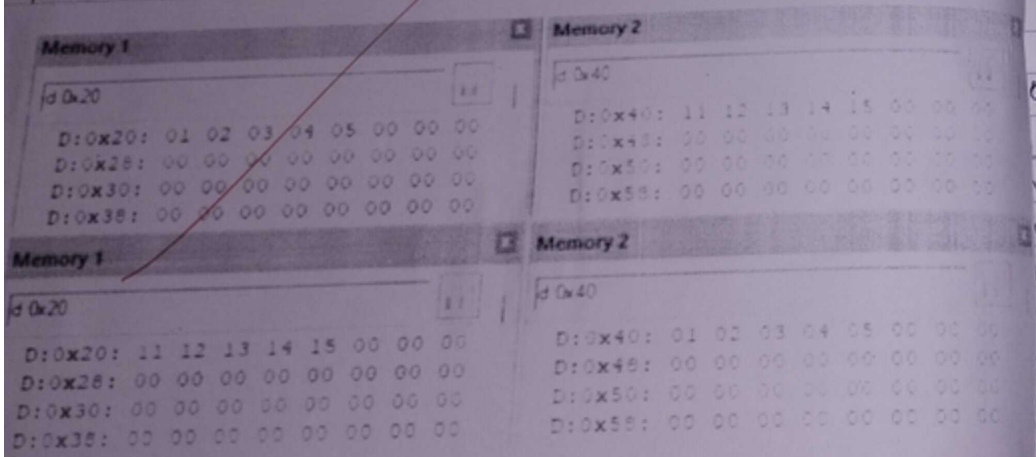
mem	data	mem	data
2000H	11	2050H	11
2001H	12	2051H	12
2002H	13	2052H	13
2003H	14	2053H	14
2004H	15	2054H	15

Thanks

EXPERIMENT-03

Algorithm:-

Step 1:- Load source address into register
Step 2:- Load destination address into another register
Step 3:- Load 3rd register with number of blocks to move.
Step 4:- Loop initialization move data pointer by address of R0 to the accumulator
Step 5:- using XCH instruction to exchange the data between accumulator & destination register
Step 6:- Move accumulator data to address pointed by source register
Step 7:- Increment stack pointer & destination register by one byte
Step 8:- Decrement & Jump target if the current register is not equal.
Step 9:- This ends the program module.



Experiment - 03

Write an ALP to exchange a source block starting with address 20H (Internal RAM) containing 5 bytes of data with destination block starting with address 40H.

Resource required :- Software Keil-vision

Program:-

```
        MOV R0, #20H
        MOV R1, #40H
        MOV R2, #05H
UP:     MOV A, @R0
        XCH A, @R1
        MOV @R0, A
        INC R0;
        INC R1;
        DJNZ R2, UP
        END.
```

Observation

Before execution

mem	data	mem	data
20H	01	40H	11
21H	02	41H	12
22H	03	42H	13
23H	04	43H	14
24H	05	44H	15

After execution

mem	data	mem	data
20H	11	40H	01
21H	12	41H	02
22H	13	42H	03
23H	14	43H	04
24H	15	44H	05

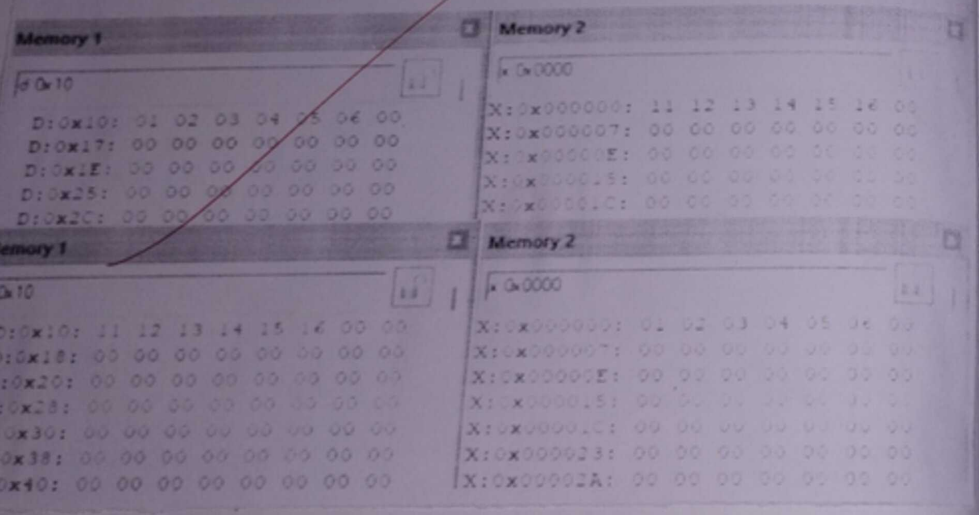
Result and conclusion :-
Exchange of address is done effectively.

Module.

EXPERIMENT - 04

Algorithm:

1. Load source address into register
2. Load another register with number of blocks to move.
3. Load data Pointer with 0000H address into another register
4. Loop initialization move data Pointer by address of data Pointer to
5. Use XCH instruction to exchange values of accumulator & source register.
6. Move accumulator data into data Pointer.
7. Increment stack Pointer of data Pointer by one byte.
8. Decrement & jump if counter register is not equal to zero.
9. Then ends the program module



Experiment no - 04

write an ALP to exchange the source block starting with address 10H internal memory, containing 6-bits of data with destination block starting at location 0000H (External RAM).

Resource required:- Software Keil-vision.

Program:-

```
MOV R0, #0H
MOV DPTR, #0000H
MOV R2, #06H
UP: MOVX A, @DPTR
    XCH A, @R0
    MOVX @DPTR, A
    INC R0
    INC DPTR
    DJNZ R2, UP
    END.
```

Observation:-

Before execution

num	data	num	data
10H	01	00H	11
11H	02	01H	12
12H	03	02H	13
13H	04	03H	14
14H	05	04H	15
15H	06	05H	16

After execution

num	data	num	data
10H	11	00H	01
11H	12	01H	02
12H	13	02H	03
13H	14	03H	04
14H	15	04H	05
15H	16	05H	06

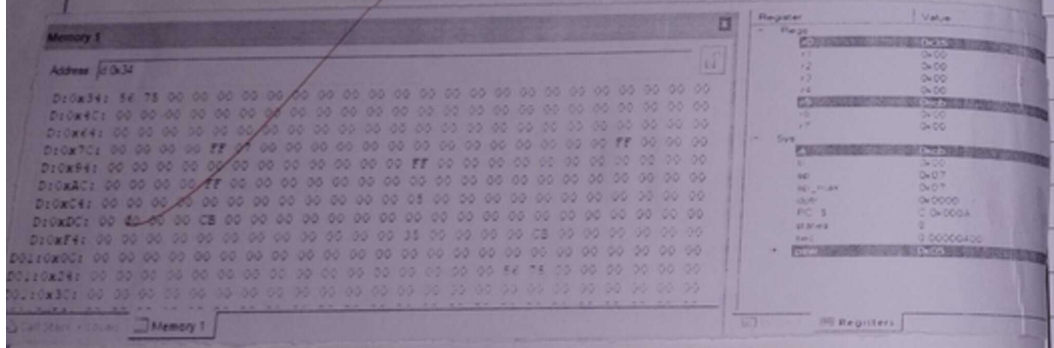
Result and conclusion.-

Exchange of address is done using ALP to external memory.

EXPERIMENT - 05

Algorithm:-

1. Load the value from memory location 34h into accumulator
2. Add the value from memory location 35h to the content of Accumulator A.
3. Store the result of the addition into register R5
4. Check for carry:-
   - If addition resulted in a carry.
   - Increment the value in R6
   - Else do nothing
5. End the Program.



Experiment - 05

Write an ALP to add the byte in the RAM add 24H and 35H store the result in the register R5 (LSB) and (MSB) using indirect addressing mode.

Resource required :- Software Keil-Mvision.

Program:-

```
mov r0, #34H
mov a, @R0
Inc R0
add a, r0
mov R5, a
jnc down
inc R6
down : nop
end
```

Observation

mem	data
34 h	54
35 h	75

Before execution (memory window)

mem	data
R5	0x29
R6	0x00

After execution (Register window)

Result and conclusion:-

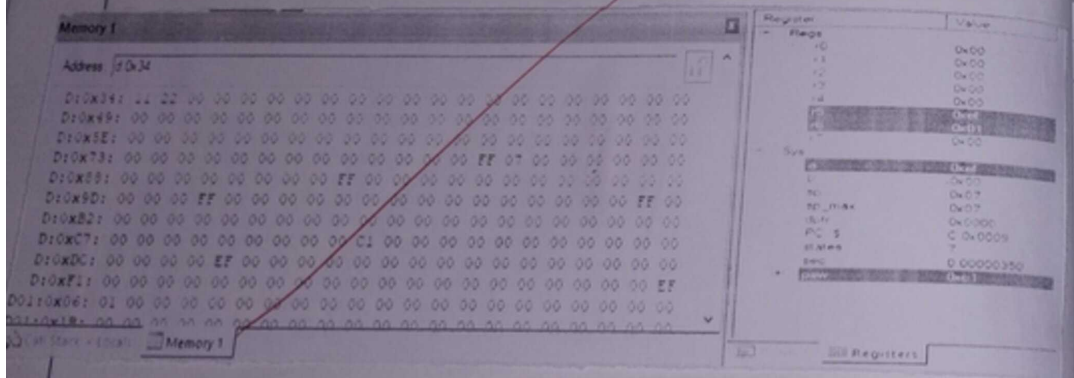
Addition of given address is done using ALP and result is stored in given registor of the RAM.

Module-

EXPERIMENT - 06

Algorithm :-

1. Load the first value from memory location 34h into accumulator.
2. Subtract the value from memory location 35h from A.
   * If there was a previous borrow (carry flag set), subtract that also.
   * This is done using the SUBB instruction.
3. Store the result in the R5.
4. Check for Borrow
   * if no borrow occured, go to step 6.
   * If a borrow occured increment the value in register R6
5. End Program.



Experiment - 06

Write an ALP to sub the byte in the internal RAM 34H and 35H. Store the result in the register R5 (LSB) & R6 (MSB).

Resource Required :- Software Keil-µvision

Program :-

```
MOV a, 34h
Subb a, 35h
mov r5, a
jnc down
inc r6
down : nop
end
```

Observation :-

memory	data
34h	11
35h	22

memory	data
R5	ef
R6	01

Result and conclusion :-

Substraction of given address is done using ALP to internal memory.

Madhu

EXPERIMENT-07

Algorithm:

Step 1:- Move load the 1st address to the accumulator

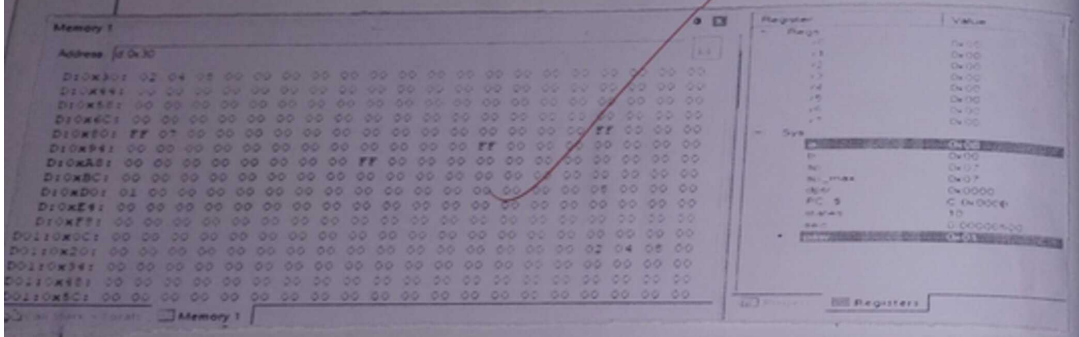
Step 2:- Load the 2nd address to the B- register

Step 3:- Multiply the accumulator and B register values

Step 4:- Move the data from accumulator to the destination address 1

Step 5:- Move the data from B register to destination address 2

Step 6:- This ends the module and execution is stopped.



Experiment - 07

Write an ALP to multiply 2 8-bit numbers stored at 30H and 31H and store 16 bit result in 32H and 33H of internal RAM.

Software used :- Keil-µ-vision5

Program:

```
MOV A, 30H
MOV B, 31H
MUL AB
MOV 32H, A
MOV 33H, B
END
```

Observation.

Before execution

Address	Data
30H	05
31H	03
32H	00
33H	00

After execution

memory	data
30H	05
31H	03
32H	0F
33H	00

Result and conclusion:

Multiplication of two 8-bit numbers at given addresses is done using ALP to internal RAM

Madhu.

EXPERIMENT-08

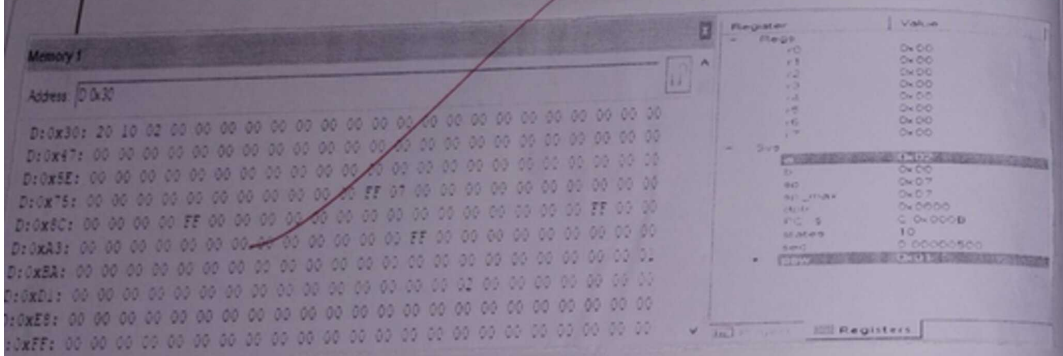
Algorithm:-

Step 1:- Initialize registers
* Set register A to 08h
* Set register B to 02h

Step 2:- Perform division

Step 3:- Result after execution
* Register A contains the quotient
* Register B contains the remainder.;

Step 4:- end



Experiment - 08

Write an ALP to perform division operation on 8-bit number by 8-bit number.

Software used :- Keil - µ-vision

Program:-

```
MOV A, #08H
MOV B, #02H
DIV AB
END
```

Observation

Before execution

num	data
a	0X2D
b	0X10

After execution

num	data
a	0X20
b	0X00

Result and conclusion:-

Division is done using ALP on 8-bit number.

Madhu.

EXPERIMENT - 09

Algorithm:

1. Load the address into a reg
2. Load the no of data blocks to a new reg
3. Loop initialization move the source address to the acc.
4. Using inst RLC route the acc to left with carry.
5. Jump if no carry is generated to a new target
6. Increment the reg R2
7. Jump to the loop back 2
8. Loop back1, increment the reg R2
9. Loop back2, increment the source reg
10. end the program.

Observation

Before execution

20h	00
21h	00
22h	00
23h	00
24h	00

40h	00
41h	00
42h	00
43h	00
44h	00

0X 2000	FE
2001	31
2002	F4
2003	12
2004	A2

After execution

20h	31
21h	12
22h	00
23h	00
24h	00

40h	FE
41h	F4
42h	A2
43h	00
44h	00

2000	FE
2001	31
2002	F4
2003	12
2004	F2

Experiment - 09

write an ALP to seperate positive & negative in a given array.

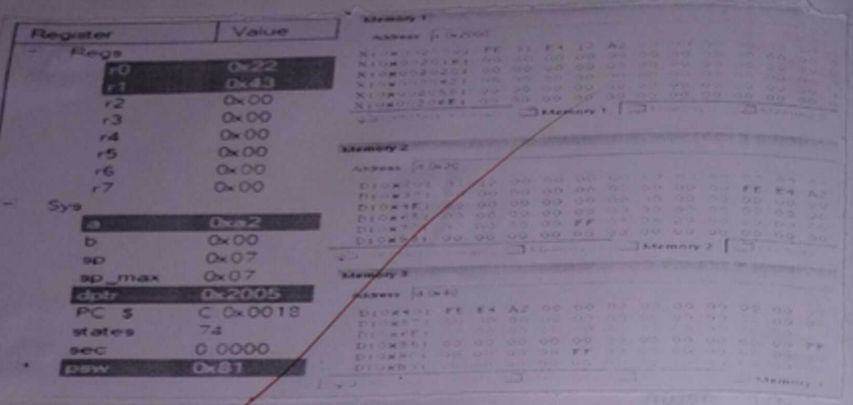
Software used: Keil- µ-vision 5

Program:-

```
mov r0, #20h
mov r2, #05h
mov r1, #40h
mov dptr, #2000h
up: mov a, @r0
rlc a
jnc down
mov a, @r0
mov @r1, a
inc r1
jmp skip
down: mov a, @r0
mov x @dptr, a
inc dptr
skip: inc r0
djnz r2, up
end
```

Result and conclusion:

Hence, the seperation of positive and negative numbers is done using ALP in a given memory.



EXPERIMENT-10

Algorithm:-

1. Load the address into a reg
2. Load the no of data blocks to a new reg
3. Loop initialization move the source address to the acc
4. using init RRC, rotate the acc to right with carry
5. Jump if no carry is generated to a new target
6. Increment to reg R2
7. Jump to the loop back 2
8. Loop back 1, increment the reg R2
9. Loop back 2, increment the source reg
10. End the program

Observation:-

Before execution

X:0X002000	D:0X20	D:0X40
13	00	00
33	00	00
24	00	00
44	00	00
12	00	00

After execution

X:0X2000	D:0X20	D:0X40
13	44	13
33	24	33
44	12	00
24	00	00
12	00	00

Experiment – 10

Write an ALP to seperate even or odd elements in the given array.

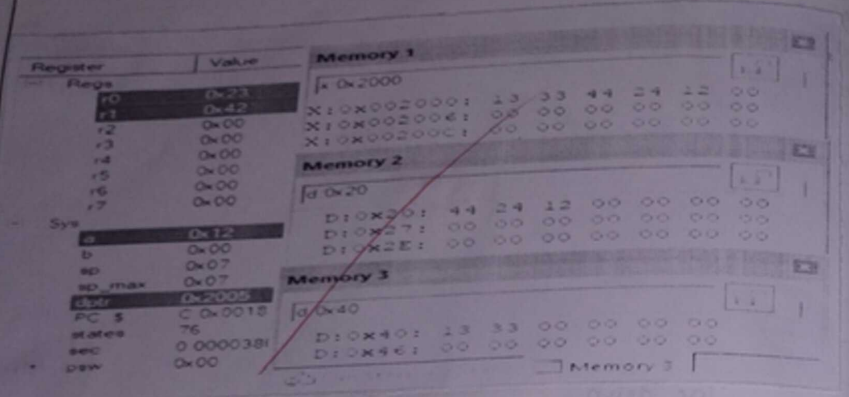
Software used:-

Program:-

```
        mov r0, #20h
        mov r2, #05h
        mov r1, #40h
        mov dptr, #2000h
up:     mov a, @r0
        rrc a
        jnc down
        mov a, @r0
        mov @r1, a
        inc r1
        jmp skip
down:   mov a, @r0
        mov x @dptr, a
        inc dptr
skip:   inc r0
        djnz r2, up
        end.
```

Result and conclusion:-

Hence, seperation of even and odd numbers is done in ALP using the required instructions in a given memory.



EXPERIMENT - 11

Algorithm:-

1. Initialize Registers → let r_3 = 4h, r_0 = 20h, r_2 = 4h
2. Start loop → move data from 40h to @ r0 then increase
3. check condition → compare a with 40h, if unequal jump to skip or use JNC
4. modify data → If carry is set, skip; other wise, store A at @
5. Adjust memory → Decrement r0, store A, then increment r0 again
6. Skip section → manage conditional jumps
7. Repeat inner loop → Decrease r_2, loop until zero
8. Repeat outer loop → Decrease r_3, loop until zero
9. end of execution.

Observation:-

Ascending

Before execution	After execution
45	08
08	09
65	15
09	45
15	65

Descending

Before execution	After execution
05	34
14	26
26	14
06	06
34	05

Experiment – 11

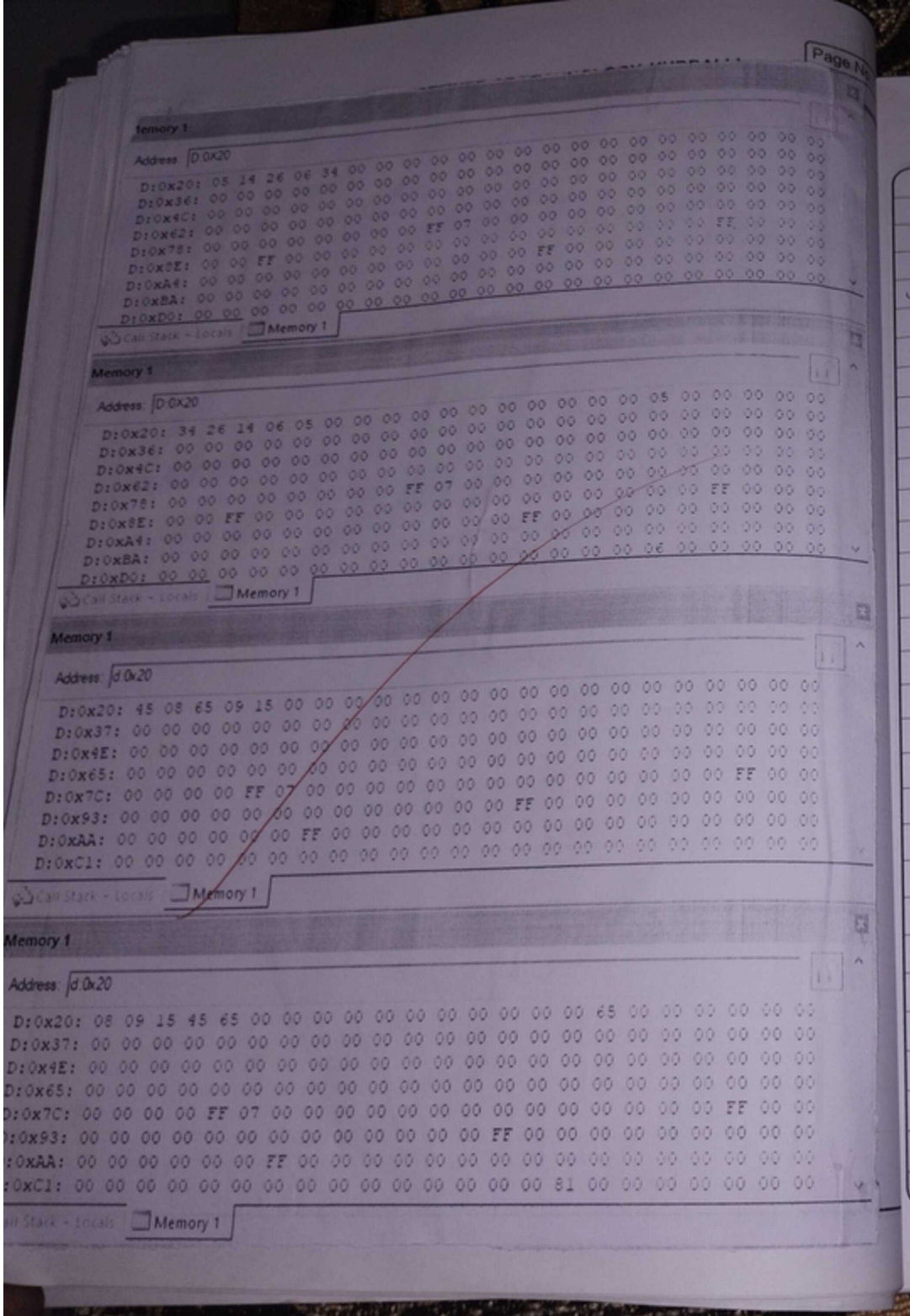
Write an ALP to arrange the numbers in ascending and descending order.

Resource Required :- Keil-µ-vision 5.

Program:-

Ascending:-

```
    mov r3, #04h
up1: mov r0, #20h
     mov r3, #04h
up:  mov 40h, @r0
     inc r0
     mov a, @r0
     cjne a, 40h, down
     jmp skip
down: jnc skip
     mov @r0, 40h
     dec @r0
     mov @r0, a
     inc r0
skip: djnz r2, up
     djnz r3, up1
     end
```



K.L.E. INSTITUTE OF TECHNOLOGY, HUBBALLI.

Page No.: 30
Date :

Program:

descending:

```
mov r3, #04h
up1: mov r0, #20h
     mov r2, #04h
up:  mov 40h, @r0 //20h
     inc r0//22h
     mov a, @p0
     cjne a, 40h, down
     jmp skip
down: jc skip
     mov @r0, 40h//23h
     dec r0//20
     mov @r0, a//20
     inc r0//21h
skip: djnz r2, up//3
     djnz r3, up1
     end
```

Result and conclusion:-

Hence, arrangement of numbers in ascending and descending order is done using an ALP.

EXPERIMENT - 12

Algorithm:

1. Initialize register → set R0: 20h and R2 = 04h
2. Store initial value → move data from 40h to @R0
3. Start loop → Increment R0, load value from @R0 into A
4. Compare value → check if A = = 40h ; if not jump to down or INC for smallest number
5. Handle Data → If carry flag is set, skip modification otherwise store A in 40h
6. Skip section → controls the program flow based on conditions
7. Repeat loop → decrease R2, repeat until zero.
8. End execution → stop program after all interaction.

Observation:

Largest:

Before execution

20h	FF
21h	FE
22h	2F
23h	61
24h	CA

After execution

20h	FF
21h	00
22h	00
23h	00
24h	00

Smallest

Before execution

20h	A3
21h	7F
22h	2B
23h	99
24h	EF

After execution

20h	2B
21h	00
22h	00
23h	00
24h	00

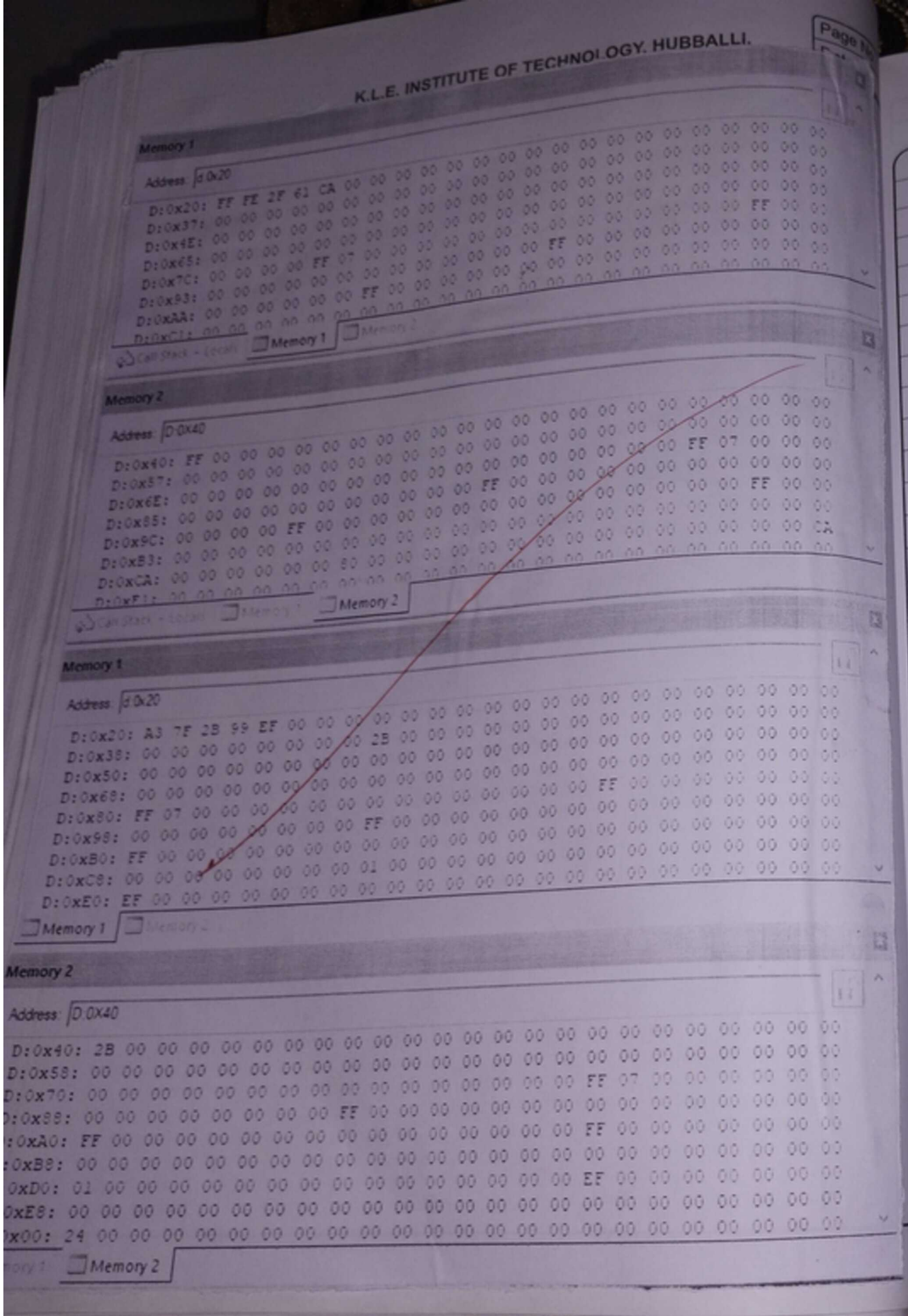
Experiment no - 12

Write an ALP to find largest and smallest number from a given array storing from 20h and store it in internal memory location 40h

Resource required:- Software Keil-µvision 5

Program:- For large number

```
        MOV R0, #20h
        MOV R2, #04h
        MOV 40H, @R0
up:     INC R0
        MOV A, @R0
        CJNE A, 40H, down
        JMP skip
down:   JC skip
        MOV 40H, A
skip:   DJNZ R2, up
        END.
```



Program :- for small number

```
        mov R0, #20H
        mov R2, #40H
        mov 40H, @R0
up:     inc R0
        mov A, @R0
        cjne A, 40H, down
        jmp skip
down:   jnc skip
        mov 40H, A
skip:   djnz R2, up
        END.
```

Result and conclusion

The ALP correctly identified the largest and smallest number from the given array stored in internal RAM starting from address 20H and stored result at memory location 40H.

EXPERIMENT - 13

Algorithm:

1. Set loop counter (r_2) to 255
2. Initialize memory location 30h to 0
3. call 'delay' subroutine to create a time delay
4. Increment the value in the memory location 30h
5. call delay subroutine again
6. Decrement the loop counter (r_2)
7. If (r_2) is not equal to zero (more loops) jump back to step3
8. The program ends (ret)

Observation:

Before execution :-

memory	data
030H	034
080H	000
081H	000
090H	000

After execution

memory	data
030H	034
080H	255
081H	009
090H	255

Experiment no - 13

Write an ALP for decimal up counter

Resource required :- software recil-M-vision 5

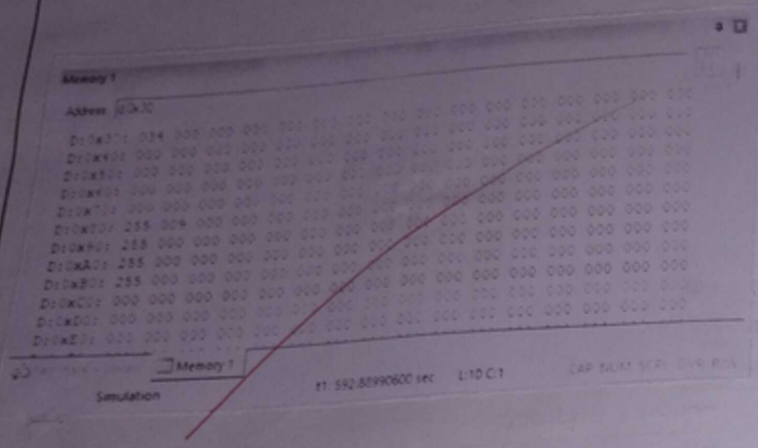
Program:

```
mov r2, #255
mov 30h, #00h
A CALL DELAY
up: inc 30h
A CALL DELAY
djnz r2, up
delay : mov r3, #255
here1 : mov r2, #255
here2 : mov r1, #255
here3 : djnz r1, here3
djnz r2, here2
djnz r3, here1
ret
end
```

Result and conclusion:

Decimal up counter is performed using ALP It increments the value at memory location from 00 to FF in decimal format with appropriate delays.

T Abdulla



EXPERIMENT – 14

Algorithm:

1. Set the loop counter (r2) to desired iterations
2. Set counter in 30h to starting value (255)
3. up: lable
   - decrement 30h (Counts down)
   - call delay
4. decrement loop counter (r2)
5. Jump to up if (r2) is not zero (repeat loop)
6. use the final value in 30h
7. Program ends

Observation:

Before execution

memory	data
080H	245
080 H	00
021 H	00
090H	00

After execution

memory	data
030H	245
080 H	255
081H	009
090H	255

Experiment no - 14

write an ALP for decimal down - counter.

Resources required : software - keilµvision 5

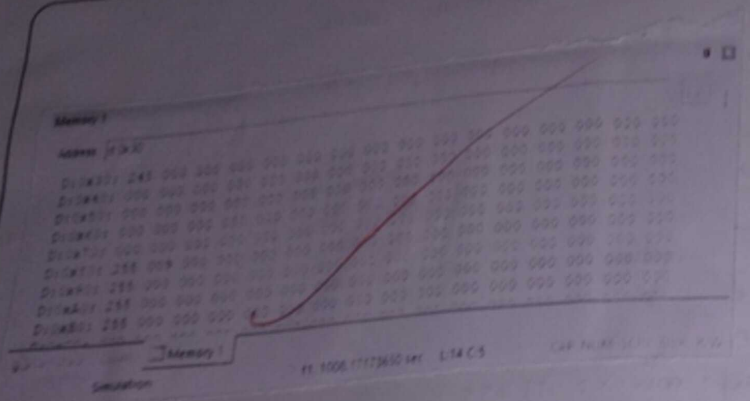
Program:-

```
mov r2, #255
mov 30h, #255
A CALL DELAY
up: dec 30h
A CALL DELAY
djnz r2,up
delay: mov r3, #255
here1 : mov r2, #255
here 2 : mov r1, #255
here 3: djnz r1, here 3
djnz r2, here 2
djnz r3, here 1
ret
. end
```

Result and conclusion:-

Decimal down counter is performed using ALP It decrements the value of FF to 00 in decimal format with delays for each step.

Page No.
Date :



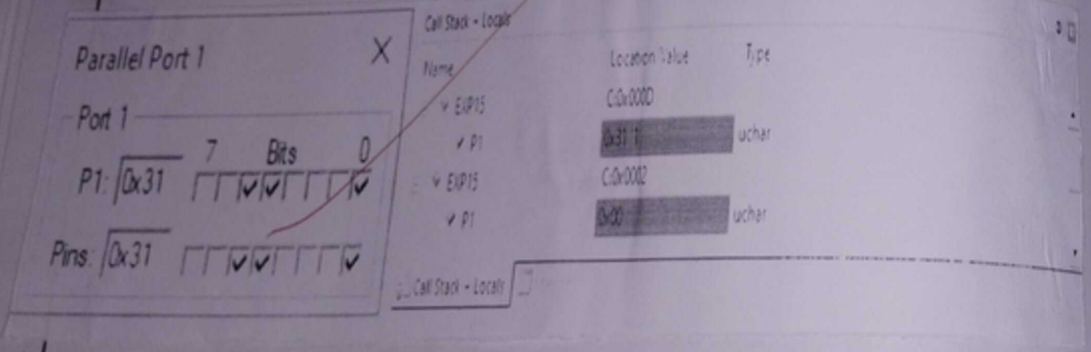
Page No. : 33
Date :

EXPERIMENT - 15

Page No.:
Date :

Algorithm:-

1. Set loop counter (r2) to the desired number of iterations
2. Set the initial value for Port P1 (00 for initial)
3. up label
   - Increment the value in Port P1
   - Call the delay subroutine to create time delay
4. Decrement the loop counter
5. Jump back to the up label if r2 is not zero (repeat loop for remaining iterations)
6. Program end (ret)



Page No.: 39
Date :

Experiment no - 15

write an ALP for hexadecimal up counter

Resource required :- software - Keil-µ-vision 5

Program:-

```
mov r2, #255
mov P1, #00H
a call delay
up: inc P1
a call delay
djnz r2, up
delay: mov r3, #255
here1: mov r2, #255
here2: mov r1, #255
here3: djnz r1, here3
djnz r2, here2
djnz r3, here1
ret
end
```

Result and conclusion:-

The counter incremented from 00H to FFH in hexadecimal format displaying each value with delays.

Teacher

Page No.:
Date :

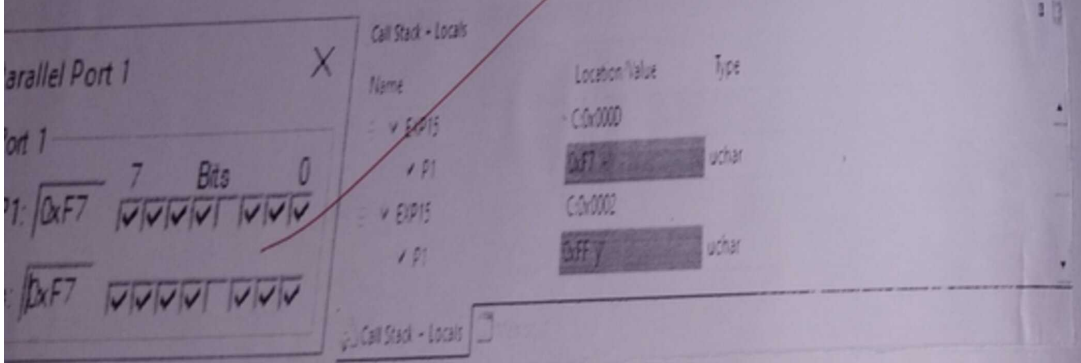
K.L.E. INSTITUTE OF TECHNOLOGY, HUBBALLI

Page No.: 110
Date :

EXPERIMENT-16

Algorithm:

1. Set loop counter (r3) to desired number of iterations.
2. Set all output pins on PORT P1 to high
3. up: label
   - Decrement the value in Port P1. This likely turns off LSB or an output pin connected to P1
   - Call the delay subroutine to create time delay.
4. Decrement the loop counter (r2).
5. Jump back to the up label if r2 is not zero (repeat loop, turning off another output pin on P1 with each loop)
6. Program ends (ret)



Experiment no - 16

Write an ALP for Hexadecimal down-counter

Resource required: Software - Keil uVision5

Program:-

```
mov r2, #255
mov P1, #255
a call delay
up: dec P1
a call delay
djnz r2, up
delay: mov r3, #255
here1: mov r2, #255
here2: mov r1, #255
here3: djnz r1, here3
djnz r2, here2
djnz r3, here1
ret
end
```

Result and conclusion:-

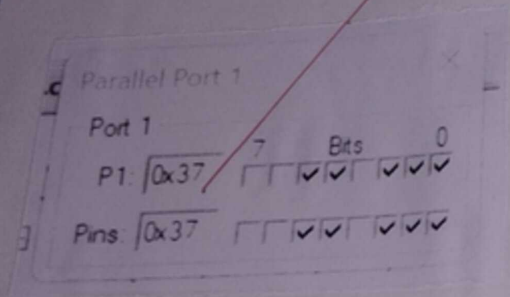
The Program counted down from FFH to 00H in hexadecimal format with delays.

EXPERIMENT - 17

Page No.:
Date:

Algorithm:-

1. Declare the header files that are to be used
2. Declare two integers variables : i.e loop counters & sum (to store integers)
3. Initialize i to 1 & sum to 0
4. Use a for loop to iterate to items (1–10)
5. Inside loop:- Add current value of i to the sum variable
6. Move the sum value to a new variable & end the Program



Page No.: 45
Date:

Experiment no – 17

Write an 8051 C program to find the sum of first 10 integers.

Resource required:-

Program:-

```
#include <reg51.h>
char i, sum=0
void main()
{
  for (i=1; i<=10; i++)
  {
    sum = sum+i;
  }
  P1 = sum;
}
```

Result and conclusion:-

The C program calculates the sum of the first 10 integers demonstrating basic arithmetic operations in 8051, yielding a proper result.

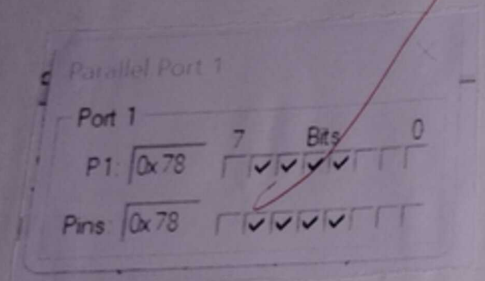
Page No.:
Date :

Page No.: 64
Date :

EXPERIMENT - 18

Algorithm:

1. Declare the header files that are to be used.
2. Declare a character & function named factorial that takes input & output in integer format.
3. Use a for loop to iterate from upto the entered number (num).
4. within the for loop, multiply the current loop counter (i) with current value of factorial.
5. update the factorial result to a new variable & Print it.
6. The Program reaches the end, signifying termination.



Experiment no - 18

write an 8051 c Program to find factorial of a given number

Resource required - software - keil-µ-vision 5

Program:

```
#include <reg51.h>
char i, fact = 1;
void main()
{
  for(i=1; i<=5; i++)
  {
    fact = fact *i;
  }
  P1 = sum;
}
```

Result and conclusion:

The Program demonstrates calculation of factorial of a given number and yields a proper result.

EXPERIMENT-19

Page No.:
Date:

Algorithm:

1. declare the required header files to be used
2. declare two assigned integer & other two variables for hexadecimal values
3. look up table declaration, An array is declared outside main program as look up table.
4. INside main function:
   - a (accumulator) - used for arithmetic ops
   - b - used to store result of squared ops
   - i- initiazed with 20h base address for look up table in memory
   - j - another loop counter intiazed with 29h
5. for loop iterates from 20h to 29 10 times
6. The program reaches to end signifying termination.

observation

Before execution

memory	data
20H	1
21H	2
22H	3
23H	4
24H	5
25H	6
26H	7
27H	8
28H	9
29H	10

memory	data
40H	000
41H	000
42H	000
43H	000
44H	000
45H	000
46H	000
47H	000
48H	000
49H	000

Page No.: 47
Date:

Experiment no - 19

Write an 8051 c Program to find the square of a number (1 to 10) using look up table

Program:

```
#include <reg51.h>
#include <absacc.h>
char i = 0x20 , j = 0x40 , a,b;
void main ()
{
  for (i= 0x20; i<= 0x29 ; i++)
  {
    a = DBYTE [i];
    b = a*a;
    DBYTE [j] = b;
      j++;
  }
}
```

Result and conclusion:-

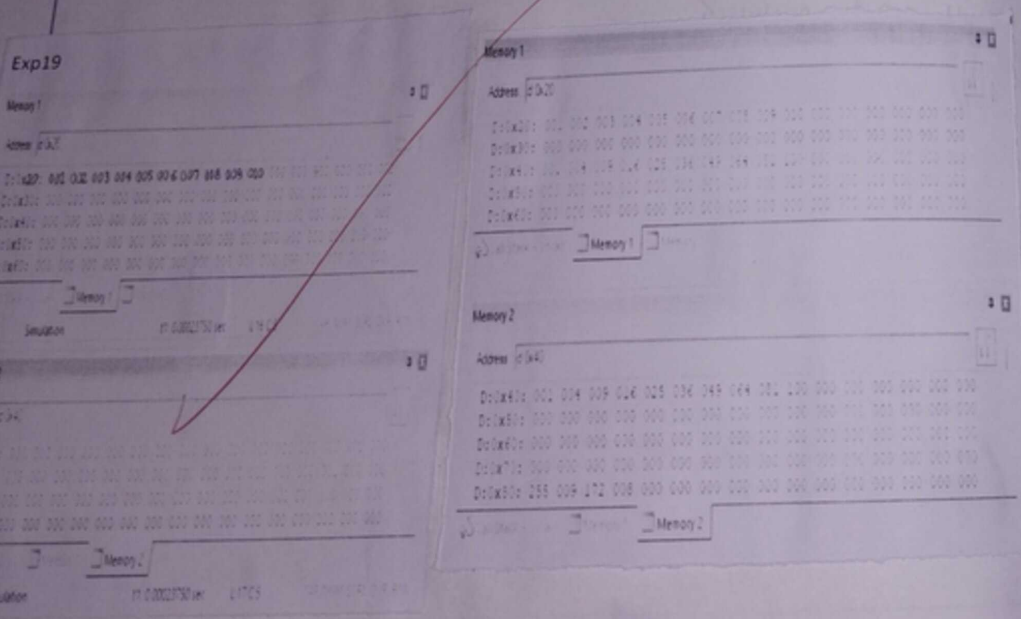
The Program calculate the square of a number using a lookup table and DBYTE macro, demonstrating efficient data retrival and calculation in 8051 c programming.

Page No.:
Date :

After execution:

memory	data
20H	1
21H	2
22H	3
23H	4
24H	5
25H	6
26H	7
27H	8
28H	9
29H	10

memory	data
40H	001
41H	004
42H	009
43H	016
44H	025
45H	036
46H	049
47H	064
48H	081
49H	100



Page No.: 43
Date :

EXPERIMENT - 20

Page No.:
Date :

Algorithm:

1. declare and call the header files that are required.
2. declare variable is loop counters used to iterate the 8-bits of main memory location.
3. zeros and ones are counter to store no. of zeros and ones.
4. 'a-ptr' is a 'char' type or pointer variable used to access memory location
5. clear array flag or PSW
6. Loop to process the 16 mem location.
7. The program ends.

Observation:

Before execution

40H	100
41H	101

After execution.

42H	007
43H	009

Page No.: 49
Date :

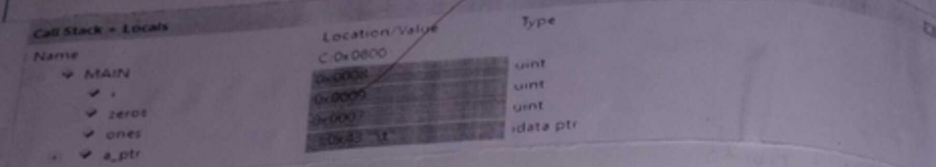
Experiment no - 20

write an 8051 c- program to count the number of ones and zeroes in 2 consecutive memory.

software used

Program:

```
include <reg51.h>
include <absacc.h>
bit carry = PSW^7;
void main ()
{
unsigned int i, zeroes, ones;
char idata * a-ptr;
a-ptr = (char*) 0x20;
P1 = 0x000;
P2 = 0x000;
P1 = *a-ptr;
a-ptr ++;
P2 = *a-ptr;
for (i = 0; i<8; i++)
{
P1 = P1 << 1;
if (carry == 1)
{
ones ++;
}
else
{
zeros++;
}
```



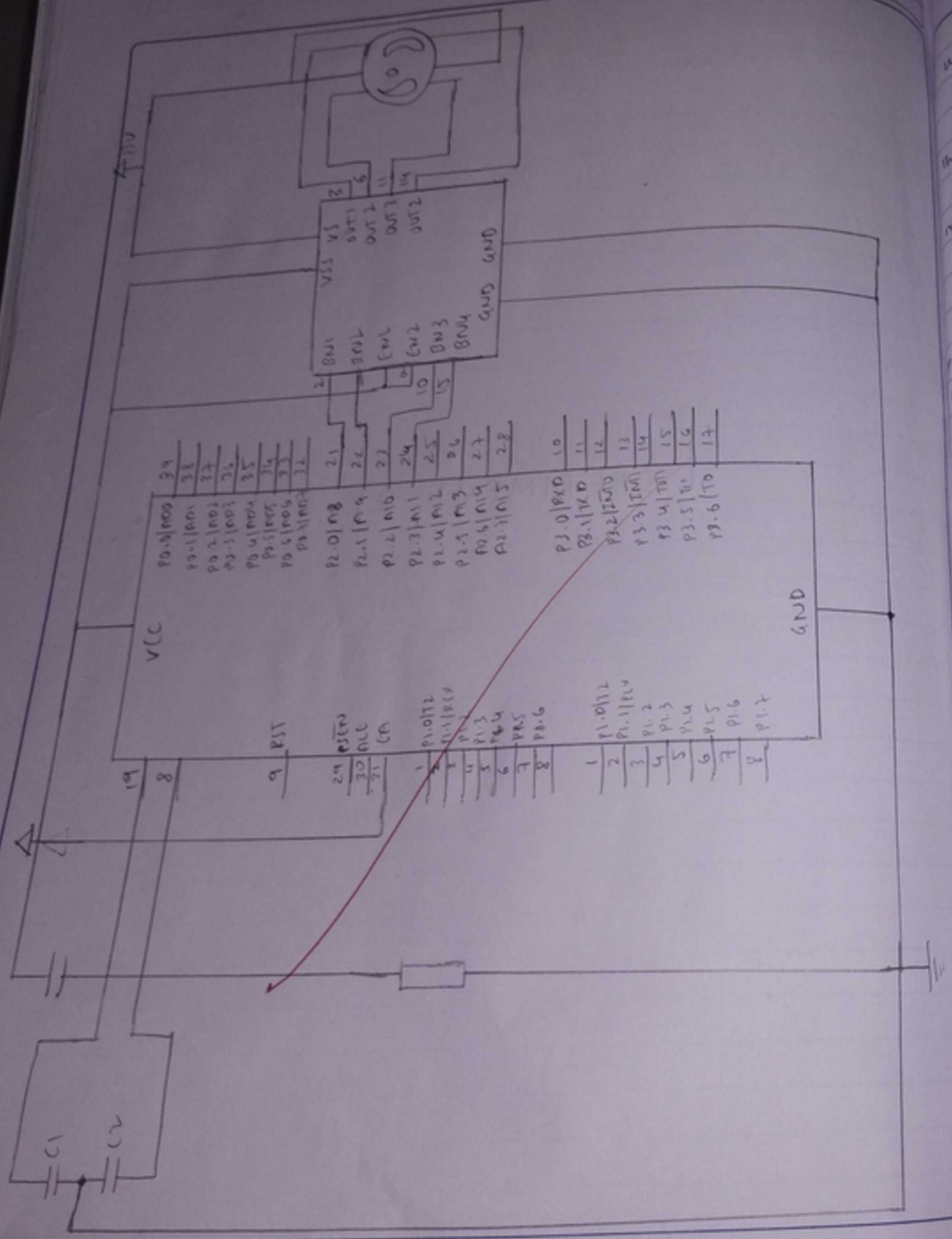

```
p2 = p2 << 1;
if (carry == 1)
{
ones ++;
}
else
{
zeroes ++;
}
}
a-ptr ++;
a-ptr = zeroes;
a-ptr ++;
a-ptr = ones;
```

Result and conclusion:

The program counts the number of ones and zeroes in two consecutive memory locations, demonstrating bit manipulation and counting techniques in 8051 C programming with accurate result.

Madhu

EXPERIMENT-21



Page No.: 52
Date:

Experiment no - 21

Write an 8051 C- program to rotate stepper motor in clock & anti- clockwise direction

Theory:-

This is the circuit diagram of driving a bipolar stepper motor using 8051 microcontroller using L293D. 24MHz crystal is connected to provide the required clock for the microcontroller 10uF capacitor & 10kΩ is used to provide power on reset (POR) for the 8051 microcontroller. L293D is connected to pins P2.0, P2.1, P2.2, P2.3 of the microcontroller and 2 pins of L293D are enabled by tieing EN1, EN2, to 5V. Logic voltage (5V) is connected to Vss pin & motor supply (12V) is connected to the Vs pin of L293D. Center tap of each windings of stepper motor is shorted and connected to the motor supply now we can energize each winding of the motor by making corresponding pin in L293D LOW.

Program:-

```
#include <reg52.h>
#include <stdio.h>
void delay (int);
void main ()
{
  do
  {
    P2 = 0X01; 110001
    delay (1000);
    P2 = 0X04 ; 110100
    delay (1000);
    P2 = 0X02 ; 110010
    delay (1000);
```

```
P2 = 0x08; 111000
delay (1000);
} while (1);

}
void delay (int k)
{
int i,j;
for (i=0; i<k ; i++)
{
   for (j=0 ; j<100 ; j++)
      }
   }
}
```

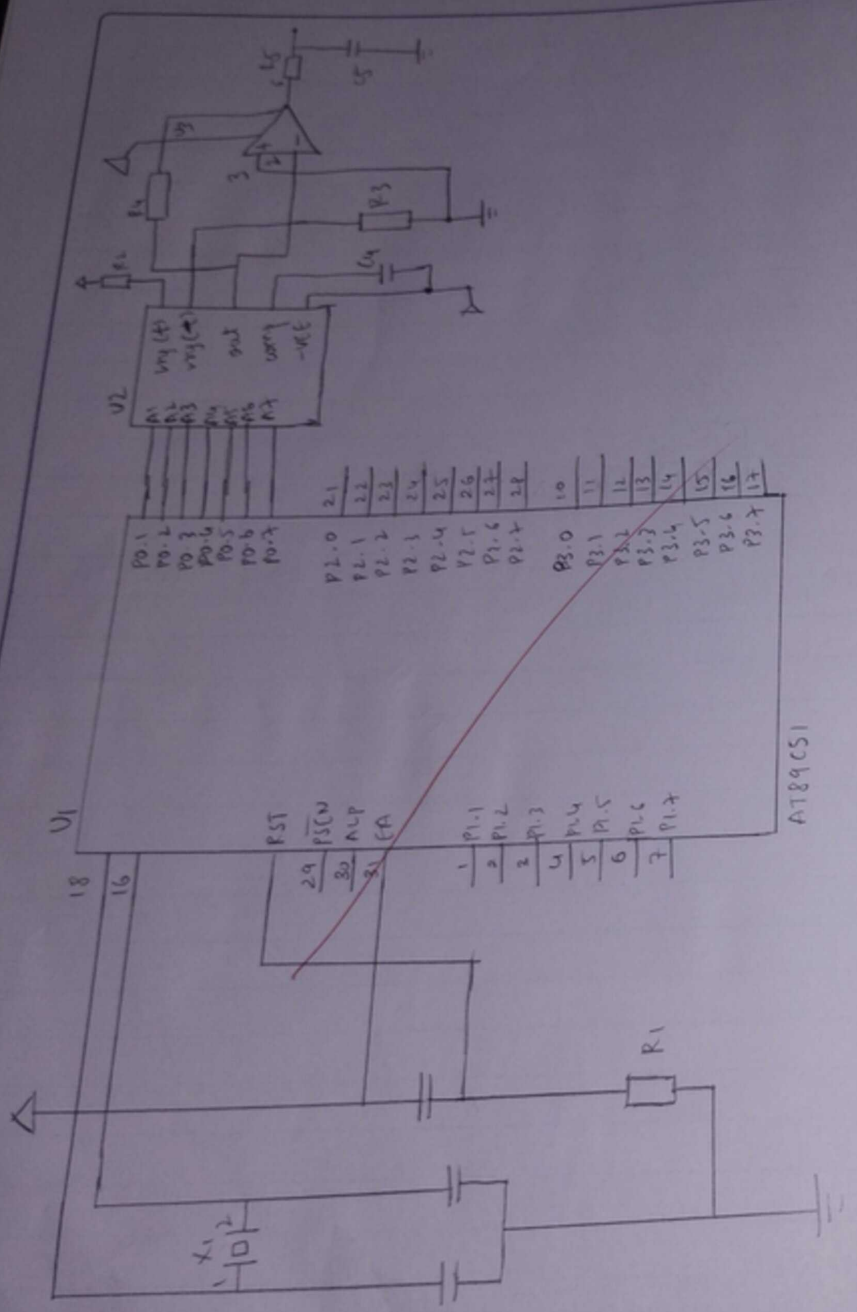
Conclusion:-

Rotate of stepper motor clock and anticlock wise using C- Program is done & verified.

K.L.E. INSTITUTE OF TECHNOLOGY, HUBBALLI

EXPERIMENT - 22

Page No.:
Date :



Page No.: 55
Date :

Experiment no - 22

Write an 8051 C Program to generate sine and square wave forms using DAC interface.

Theory:-

The digital to analog converter (DAC) is a device that is widely used for converting digital pulses to analog signals. There are two methods of converting digital signals to analog signals. These two methods are binary weighted method and R/2R ladder method. In this article we will use the MC1408 (DAC0808) digital to analog converter. This chip uses R/2R ladder method. This method can achieve a much bigger degree of precision. DACs are judged by its resolution. The resolution is a function of the number of binary inputs. The most common input counts are 8, 10, 12 etc. No of data inputs decides the resolution OR DAC. If IR there are n digital input pin, there are 2^n analog levels. For 8 input DAC has 256 discrete voltage levels.

The following formula is knowing the function OR Iout.

$$I_{out} = I_{REF}[D72 + D64 + D58 + D416 + D332 + D332 + D264 + D128 + DD256]$$

$$I_{out} = I_{REF}[D72 + D64 + D58 + D416 + D332 + D264 + D128 + DD256].$$

The I_{REF} is the input current. This must be provided into the Pin 14 - Generally 2.0MA is used as I_{REF}

Page No.:
Date :

Page No.: 56
Date :

Program:

```
# square wave
#include <reg51.h>
void delay()
{
int i=0;
for(i=0;i<922;i++)
}
void main()
{
P0 = 0XFF;
delay();
P0 = 0X00;
//sine wave
#include <reg51.h>
int main()
{
int j;
int
c[37] = {128,150,172,192,210,226,239,248,254,255,254,
248,239,226,210,192,172,150,128,106,84,64,
46,30,17,8,2,0,2,8,17,30,46,64,84,
106,128};
while(1)
{
for(j=0;j<36;j++)
{
P1 = c[j];
}
P1 = 128;
}}
```

Conclusion:- Generation of sine wave and square waveforms using one interface IC done and verified.

Madhu