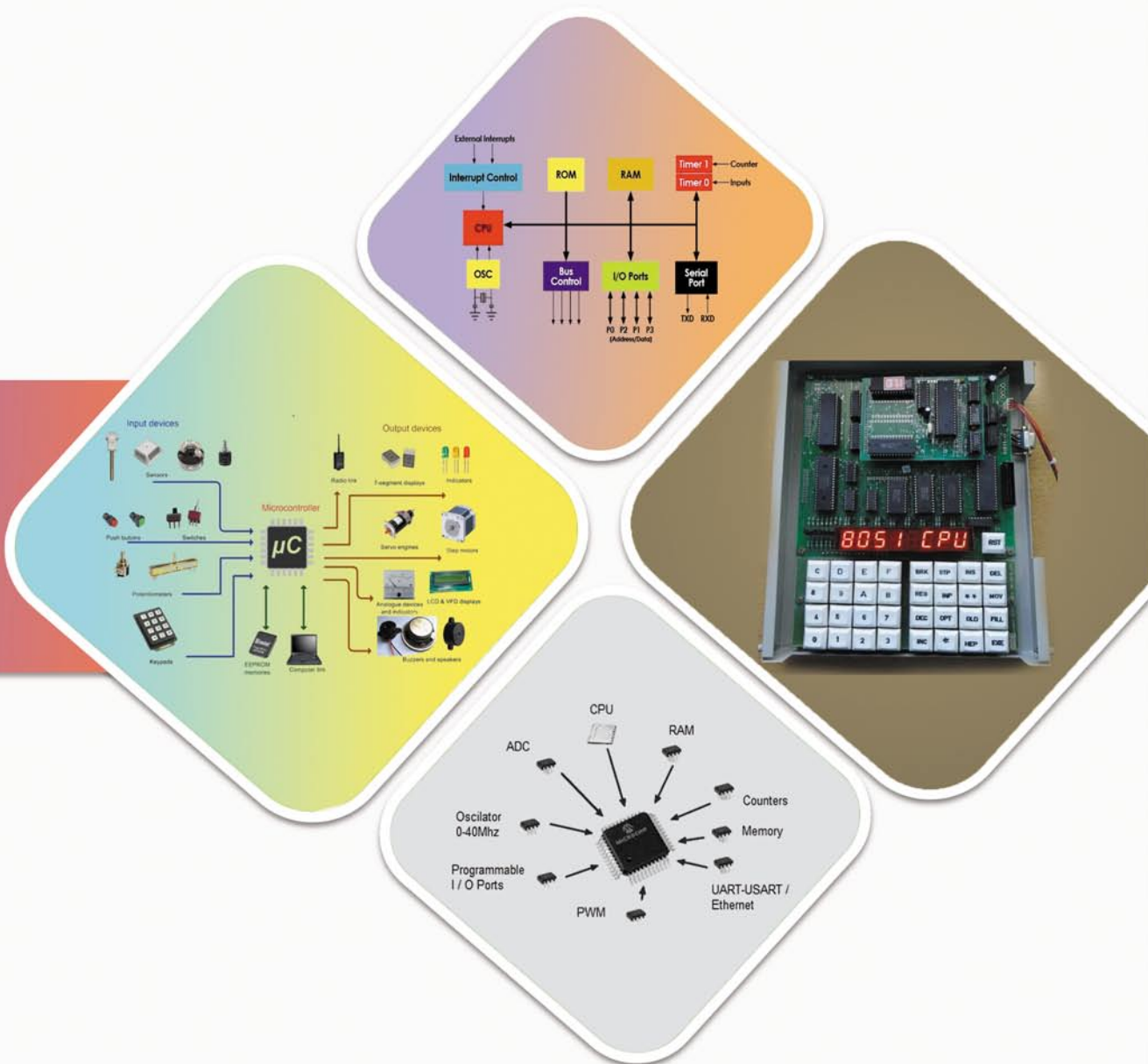


SCHEME : K

Name : _____
Roll No. : _____ Year : 20__ 20__
Exam Seat No. : _____

LABORATORY MANUAL FOR MICROCONTROLLER & APPLICATIONS (314328)



ELECTRONICS ENGINEERING GROUP



**MAHARASHTRA STATE BOARD OF
TECHNICAL EDUCATION, MUMBAI**
(Autonomous) (ISO 9001: 2015) (ISO/IEC 27001:2013)

VISION

To ensure that the Diploma Level Technical Education constantly matches the latest requirements of technology and industry and includes the all-round personal development of students including social concerns and to become globally competitive, technology led organization.

MISSION

To provide high quality technical and managerial manpower, information and consultancy services to the industry and community to enable the industry and community to face the changing technological and environmental challenges.

QUALITY POLICY

We, at MSBTE, are committed to offer the best in class academic services to the students and institutes to enhance the delight of industry and society. This will be achieved through continual improvement in management practices adopted in the process of curriculum design, development, implementation evaluation and monitoring system along with adequate faculty development Programs.

CORE VALUES

MSBTE believes in the followings:

- Education industry produces live products.
- Market requirements do not wait for curriculum changes.
- Question paper is the reflector of academic standards of educational organization
- Well-designed curriculum needs effective implementation too.
- Competency based curriculum is the backbone of need based programs.
- Technical skills do need support for life skills.
- Best teachers are the national assets.
- Effective teaching learning process is impossible without learning resources.

A Laboratory Manual for

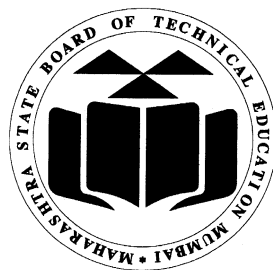
Electronics Engineering group

Microcontroller and Applications

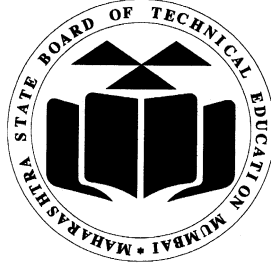
(314328)

Semester-IV

(AO,DE,EJ,EK,ET,EX,IC,IE,IS,TE)



**Maharashtra State
Board of Technical Education, Mumbai**
(Autonomous) (ISO-9001-2008) (ISO/IEC 27001:2013)



**Maharashtra State
Board of Technical Education, Mumbai**

(Autonomous) (ISO-9001-2008) (ISO/IEC 27001:2013)

4th Floor, Government Polytechnic Building, 49, Kherwadi,

Bandra (East), Mumbai – 400051.

Maharashtra State Board of Technical Education

Certificate

This is to certify that Mr. / Ms
Roll No.....of Semester of Diploma
inof
Institute.....

(Code.....) has attained pre-defined practical outcomes
(PROs) satisfactorily in course **Microcontroller and Applications**
(314328) for the academic year 20.....to 20..... as prescribed in
the curriculum.

Place

Enrollment No.....

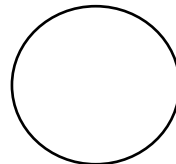
Date:

Exam Seat No.

Course Teacher

Head of the Department

Principal



Preface

The primary focus of any engineering laboratory/ field work in the technical education system is to develop the much needed industry relevant competencies and skills. With this in view, MSBTE embarked on this innovative 'K' Scheme curricula for engineering diploma programmes with outcome-based education as the focus and accumulator ordinarily, relatively large amount of time is allotted for the practical work. This displays the great importance of laboratory work making each teacher; instructor and student to realize that every minute of the laboratory time need to be effectively utilized to develop these outcomes, rather than doing other mundane activities. Therefore, for the successful implementation of this outcome-based curriculum, every practical has been designed to serve as a '*vehicle*' to develop this industry identified competency in every student. The practical skills are difficult to develop through 'chalk and duster' activity in the classroom situation. Accumulator ordinarily, the 'K' scheme laboratory manual development team designed the practical's to *focus* on the *outcomes*, rather than the traditional age old practice of conducting practical's to 'verify the theory' (which may become a by-product along the way).

This laboratory manual is designed to help all stakeholders, especially the students, teachers and instructors to develop in the student the pre-determined outcomes. It is expected from each student that at least a day in advance, they have to thoroughly read through the concerned practical procedure that they will do the next day and understand the minimum theoretical background associated with the practical. Every practical in this manual begins by identifying the competency, industry relevant skills, course outcomes and practical outcomes which serve as a key focal point for doing the practical. The students will then become aware about the skills they will achieve through procedure shown there and necessary precautions to be taken, which will help them to apply in solving real-world problems in their professional life.

This manual also provides guidelines to teachers and instructors to effectively facilitate student-centered lab activities through each practical exercise by arranging and managing necessary resources in order that the students follow the procedures and precautions systematically ensuring the achievement of outcomes in the students.

Microcontroller is used in almost all the domestic, industrial, consumer goods and other high end products. Automation is used in every field of engineering and microcontroller is inbuilt element of these systems and devices. Diploma engineers have to deal with various microcontroller based systems and maintain them. This course is intended to develop the skills to maintain and solve the application problems related to microcontrollers.

The lab manual development team wishes to thank MSBTE who took initiative in the development of curriculum re-design project and implementation and also acknowledge the contribution of individual course experts who have been involved in laboratory manual as well as curriculum development (I scheme) directly or indirectly. The National Institute of Technical Teachers' Training and Research, Bhopal deserves our sincere appreciation for the guidance provided.

Although all care has been taken to check for mistakes in this laboratory manual, yet it is impossible to claim perfection especially as this is the first edition. Any such errors and suggestions for improvement can be brought to our notice and are highly welcome.

Programme Outcomes (POs) to be achieved through Practical of this Course

Following programme outcomes are expected to be achieved through the practical of the course

- PO 1. Basic and Discipline specific knowledge:** Apply knowledge of basic mathematics, sciences and engineering fundamentals and engineering specialization to solve the engineering problems.
- PO 2. Problem analysis:** Identify and analyze well-defined engineering problems using codified standard methods.
- PO 3. Design/Development of solutions:** Design solutions for well-defined technical problems and assist with the design of system components or processes to meet specified needs.
- PO 4. Engineering tools, Experimentation and Testing:** Apply modern engineering tools and appropriate technique to conduct standard tests and measurements.
- PO 5. Engineering practices for society, sustainability and environment:** Apply appropriate technology in context of society, sustainability, environment and ethical practices.
- PO 6. Project Management:** Use engineering management principles individually, as a team member or a leader to manage projects and effectively communicate about well-defined engineering activities.
- PO 7. Life-long learning:** Ability to analyze individual needs and engage in updating in the context of technological challenge.

List of Industry Relevant Skills

The following industry relevant skills of the competency ‘**Maintain microcontroller-based systems**’ are expected to be developed in students by undertaking the practical’s of this laboratory manual.

1. Identify the relevant microcontroller.
2. Interface various I/O devices with microcontroller.
3. Interpret the program.
4. Maintain microcontroller-based systems.
5. Use features available with given microcontroller.
6. Test the circuit for the given application.
7. Find faults and trouble shoot the given system.

Practical- Course Outcome matrix

Course Outcomes (COs)						
a. Interpret architecture of 8-bit microcontrollers. b. Develop program in 8051 in assembly language for the given operations. c. Develop program using timers and interrupt. d. Interface the memory and I/O peripherals to 8051 microcontrollers. e. Maintain microcontroller-based application						
Sr No.	Laboratory Experiment / Practical Titles / Tutorial Titles	CO a.	CO b.	CO c.	CO d.	CO e.
1.	*Identification of various blocks of 8051 microcontroller development board.	√	-	-	-	-
2.	Assembly language program using various addressing modes.	-	√	-	-	-
3.	*ALP to perform arithmetic operations on 8-bit data.	-	√	-	-	-
4.	*ALP to perform arithmetic operations on 16-bit data.	-	√	-	-	-
5.	*ALP to perform addition of BCD Data.	-	√	-	-	-
6.	*ALP for series addition.	-	√	-	-	-
7.	*Array data transfer from source locations to destination locations.	-	√	-	-	-
8.	*Block exchange of data from source locations to destination locations.	-	√	-	-	-
9.	*Finding the smallest number from the given data bytes.	-	√	-	-	-
10.	Finding the largest number from the given data bytes.	-	√	-	-	-
11.	*Arranging the number in ascending order.	-	√	-	-	-
12.	Arranging the number in descending order	-	√	-	-	-
13.	*Generate delay using Timer register.			√		
14.	*Serial 8-bit data transfer on serial port.			√		
15.	LED interfacing to 8051				√	

Course Outcomes (COs)						
a. Interpret architecture of 8-bit microcontrollers. b. Develop program in 8051 in assembly language for the given operations. c. Develop program using timers and interrupt. d. Interface the memory and I/O peripherals to 8051 microcontrollers. e. Maintain microcontroller-based application						
Sr No.	Laboratory Experiment / Practical Titles / Tutorial Titles	CO a.	CO b.	CO c.	CO d.	CO e.
16.	Generating pulse and square wave using Timer delay.	-	-	-	√	-
17.	LED matrix Interfacing to 8051.	-	-	-	√	-
18.	*Seven-segment display interface for displaying the decimal numbers.	-	-	-	√	-
19.	*Relay interfacing to Microcontroller.	-	-	-	√	-
20.	*LCD interfacing to 8051 to display characters and decimal number.	-	-	-	√	-
21.	Keyboard interfacing to 8051.	-	-	--	√	-
22.	* ADC interfacing to 8051.	-	-	-	√	-
23.	*DAC Interfacing to generate the square waveform.	-	-	-	-	√
24.	DAC Interfacing to generate the triangular waveform.	-	-	-	-	√
25.	*Stepper Motor Interfacing to 8051.	-	-	-	-	√
26.	Stepper Motor Interfacing to 8051 for rotating anti-clockwise	-	-	-	-	√
27.	Water Level controller using 8051.	-	-	-	-	√
28.	Temperature Sensor Interfacing to detect and measure temperature.	-	-	-	-	√

Guidelines to Teachers

1. Teacher is expected to refer complete curriculum document and follow guidelines for implementation
2. Teacher should provide the guideline with demonstration of practical to the students with all features.
3. Teacher shall explain prior concepts to the students before starting of each practical
4. Involve students in performance of each practical.
5. Teacher should ensure that the respective skills and competencies are developed in the students after the completion of the practical exercise.
6. Teachers should give opportunity to students for hands on experience after the demonstration.
7. Teacher is expected to share the skills and competencies to be developed in the students.
8. Teacher may provide additional knowledge and skills to the students even though not covered in the manual but are expected the students by the industry.
9. Give practical assignment and assess the performance of students based on task assigned to check whether it is as per the instructions.
10. Assess the skill achievement of the students and COs of each unit.
11. At the beginning Teacher should make the students acquainted with any of the simulation software environment as few experiments are based on simulation.
12. It is desirable to paste the photo of actual experimental setup or draw block diagram of experimental setup.
13. Practical No.1 should not be consider for Practical (ESE-End Semester Exam).
14. Conduct workshops teaching students how to safely and properly dispose of electronic waste, fostering a culture of sustainability within the lab environment.
15. Introduce zero-waste challenges where students are encouraged to create projects using only recycled or repurposed components from existing e-waste, promoting creativity and resourcefulness.

Instructions for Students

1. Listen carefully the lecture given by teacher about course, curriculum, learning structure, skills to be developed.
2. Before performing the practical student shall read lab manual of related practical to be conducted.
3. For incidental writing on the day of each practical session every student should maintain a ***dated log book*** for the whole semester, apart from this laboratory manual which s/he has to ***submit for assessment to the teacher.***
4. Organize the work in the group and make record of all observations.
5. Students shall develop maintenance skill as expected by industries.
6. Student shall attempt to develop related hand-on skills and gain confidence.
7. Student shall develop the habits of evolving more ideas, innovations, skills etc. those included in scope of manual

8. Student shall refer technical magazines, IS codes and data books.
9. Student should develop habit to submit the practical on date and time.
10. Student should well prepare while submitting write-up of exercise.

Content Page

List of Practical's and Progressive Assessment Sheet

Sr No	Title of the practical	Page No.	Date of performance	Date of submission	Assessment marks (25)	Dated sign. of teacher	Remarks (if any)
1	*Identification of various blocks of 8051 microcontroller development board.						
2	Assembly language program using various addressing modes.						
3	*ALP to perform arithmetic operations on 8- bit data.						
4	*ALP to perform arithmetic operations on 16-bit data.						
5	*ALP to perform addition of BCD Data.						
6	*ALP for series addition.						
7	*Array data transfer from source locations to destination locations.						
8	*Block exchange of data from source locations to destination locations.						
9	*Finding the smallest number from the given data bytes.						
10	Finding the largest number from the given data bytes.						
11	*Arranging the numbers in ascending order.						
12	Arranging the numbers in descending order						
13	*Generate delay using Timer register.						
14	*Serial 8-bit data transfer on serial port.						
15	LED interfacing to 8051						
16	Generating pulse and square wave by using Timer delay.						
17	LED matrix Interfacing to 8051.						
18	*Seven-segment display to interface for displaying the decimal number.						
19	*Relay interfacing to Microcontroller.						

Sr No	Title of the practical	Page No.	Date of performance	Date of submission	Assessment marks (25)	Dated sign. of teacher	Remarks (if any)
20	*LCD interfacing of 8051 to display the characters and decimal numbers.						
21	Keyboard interfacing to 8051.						
22	* ADC interfacing to 8051.						
23	*DAC Interfacing to generate the square waveform.						
24	DAC Interfacing to generate the triangular waveform.						
25	*Stepper Motor Interfacing to 8051.						
26	Stepper Motor Interfacing to 8051 for rotating anti-clockwise						
27	Water Level controller using 8051.						
28	Temperature Sensor Interfacing to detect and measure temperature						
Total Marks					.		

- The practical marked as '*' are compulsory,
- Column 6th marks to be transferred to Performa of CIAAN-2017.

Practical No. 1: Identification of various blocks of 8051 microcontroller development board

I. Practical Significance

Microcontroller has wide application in electronic system needing real time processing/control, starting from domestic application such as washing machine, TV and air conditioners. They are also used in automobiles, process control industries, cell phones, robotics and in space application.

II. Industry/Employer Expected Outcome(s)

Maintain microcontroller-based systems.

III. Course Level Learning Outcome(s)

Interpret architecture of 8-bit microcontrollers.

IV. Laboratory Learning Outcome(s)

Identify the functions of various blocks of 8051 microcontroller development board

V. Relevant Affective Domain related outcome(s)

1. Demonstrate working as a leader/a team member.
2. Maintain tools and equipment.
3. Follow ethical practices.

VI. Relevant Theoretical Background

Microcontroller is a single chip microcomputer made through VLSI fabrication. 8051 is the first microcontroller of the MCS-51 family introduced by Intel Corporation.

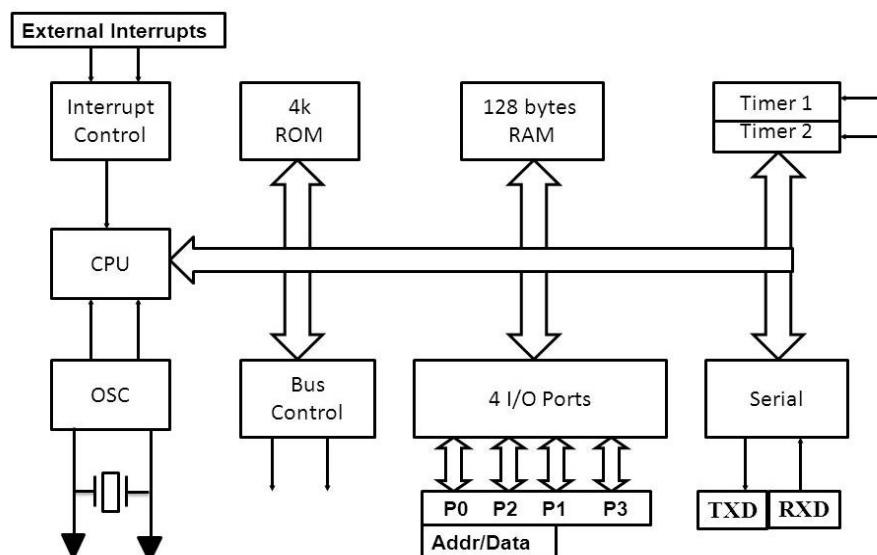


Fig 1.1 Block diagram of 8051

It has inbuilt components such as CPU, internal RAM and ROM, timers/counters, serial ports, interrupts and I/O ports. AT80C51 is compatible with MCS-51. The 8051 is an 8-bit processor. The CPU can work on only 8 bits of data at a time.

Specification of microcontroller 8051

1. It is an 8-bit microcontroller
2. It has 128 bytes of Internal RAM
3. It has 4 kilo bytes of Internal ROM
4. It has two 16-bit internal timers/counters
5. It has four 8-bit parallel ports
6. It has one full duplex serial port
7. It has three internal and two external interrupts.

Pin diagram of 8051: 8051 is a 40 pin IC and operates on +5 volts DC supply.

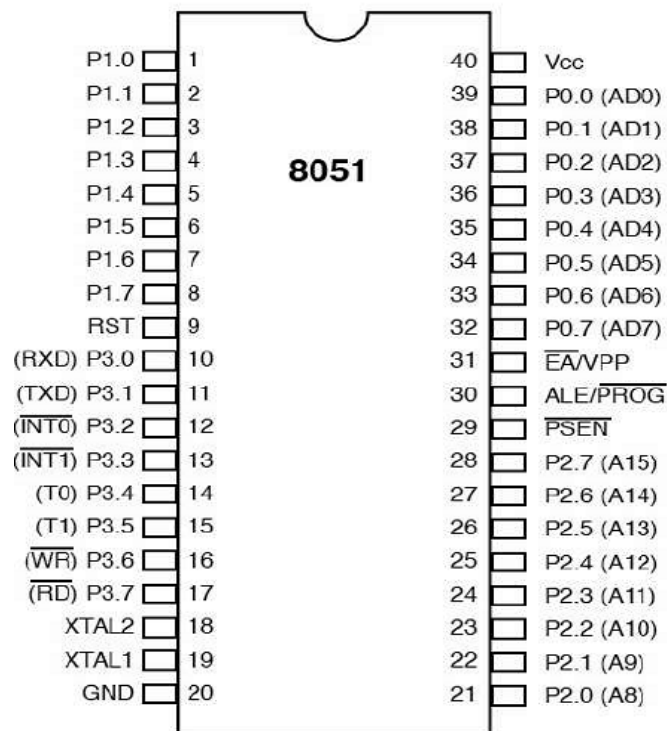


Fig. 1.2 Pin diagram of 8051

Keil IDE

Keil is 8051 development tool which includes a text Editor, Assembler Debugger, linker, Simulator, C-compiler, hex converter, locator and some in-built features like logic analyzer to observe various waveforms. It also includes terminal emulator. Keil supports all 8051 derivatives and valuable tool for embedded software development.

The Development board

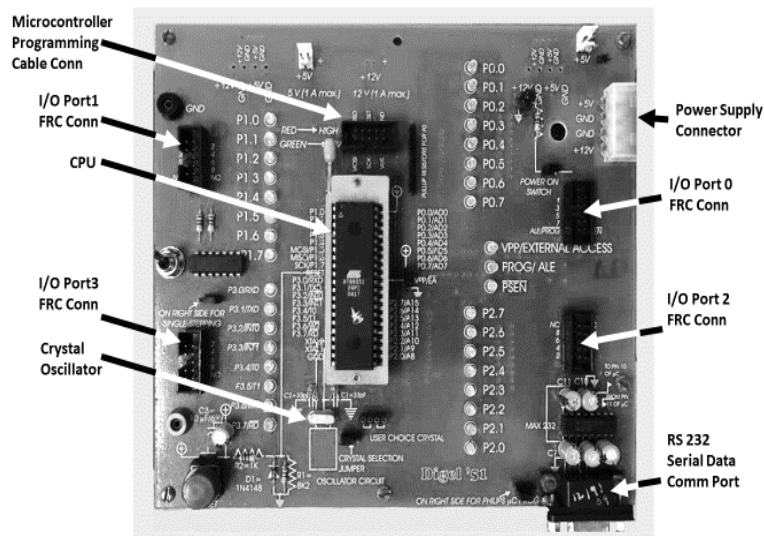


Fig. 1.3 8051 Development Board

The development board has 8051 microcontrollers along with some necessary component like MAX 232, resistor network etc. It is a devise used to develop and design a prototype embedded system. Port pins are taken out for interfacing various peripherals. It has a provision to download the hex file of user program which is generated by Keil or any other IDE. FLASH MAGIC software is used to download the hex file into the code memory of microcontroller.

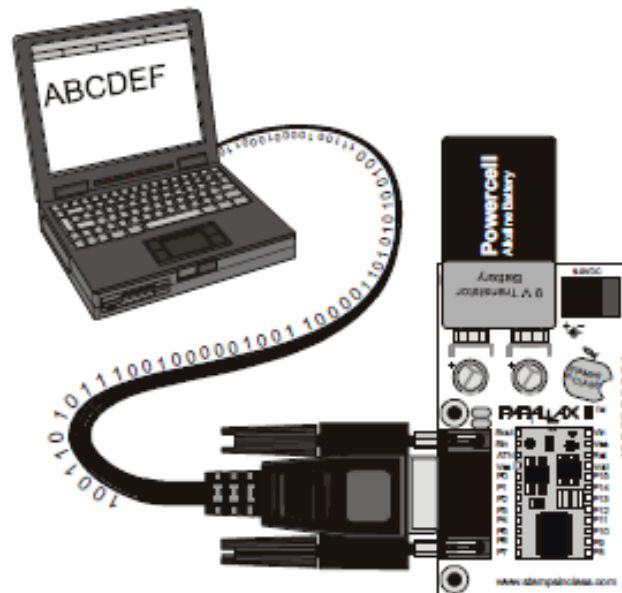


Fig. 1.4 8051 Programming through serial cable

VII. Resources required

Sr. No.	Instrument /Components	Specification	Quantity
1.	Microcontroller kit	Single board system with 8K RAM,ROM memory with battery backup,16X4,16X2LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross-c-compiler,RS-232,USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open-source IDE, simulation and program downloading software.	1 No.

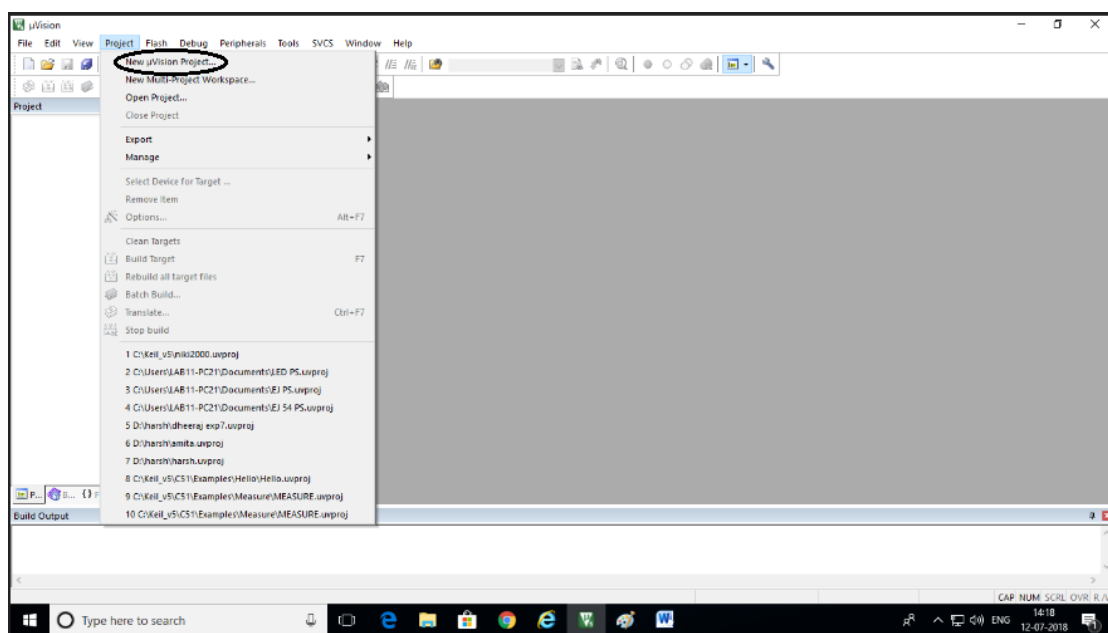
VIII. Precautions to be Followed

Do not power up development board when identifying block.

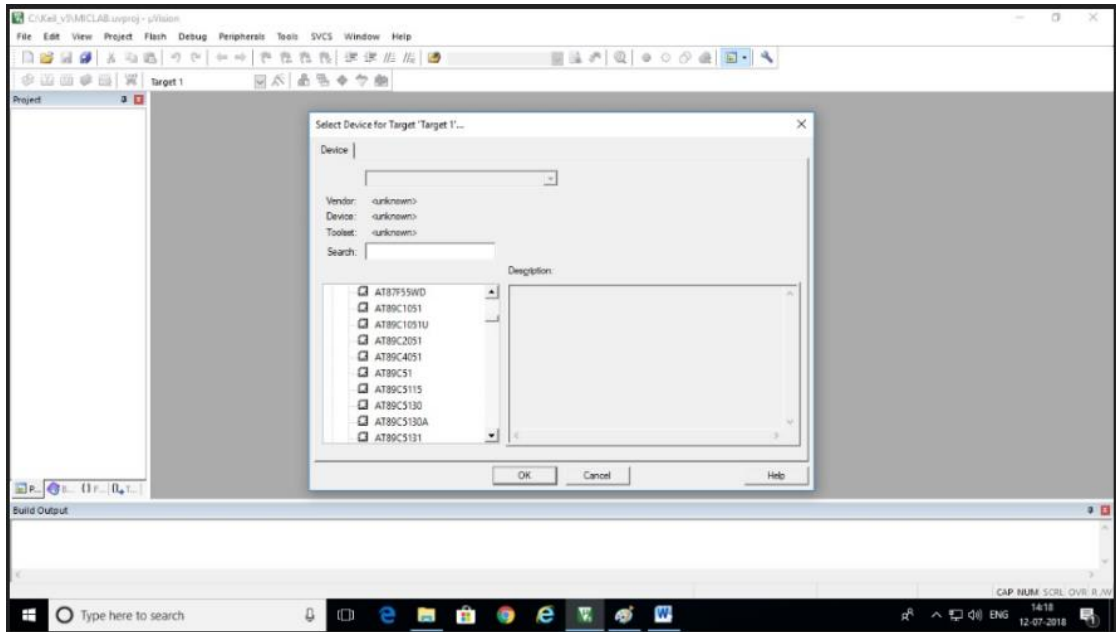
IX. Procedure

Steps for creating a project using Keil software:

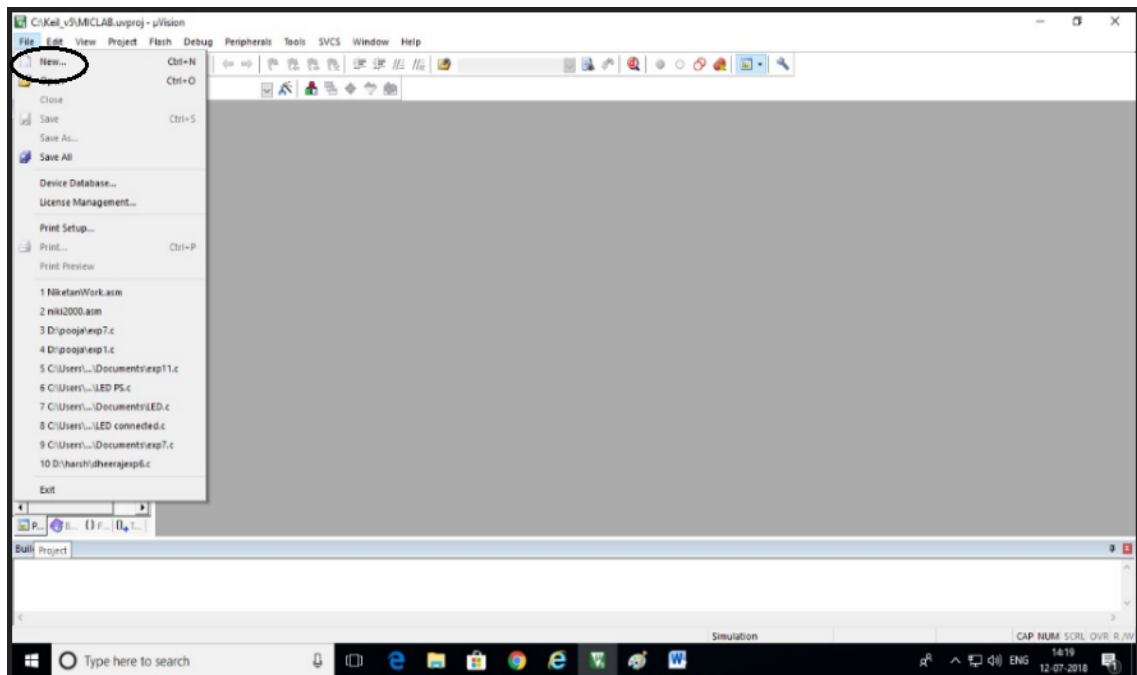
1. Start Keil by double clicking on Keil icon. (Keil automatically opens the last project which was opened previously, when Keil was closed).
2. To create new project, Click on Project and select new project.
3. Select appropriate location for new project and type project name, click on save button.



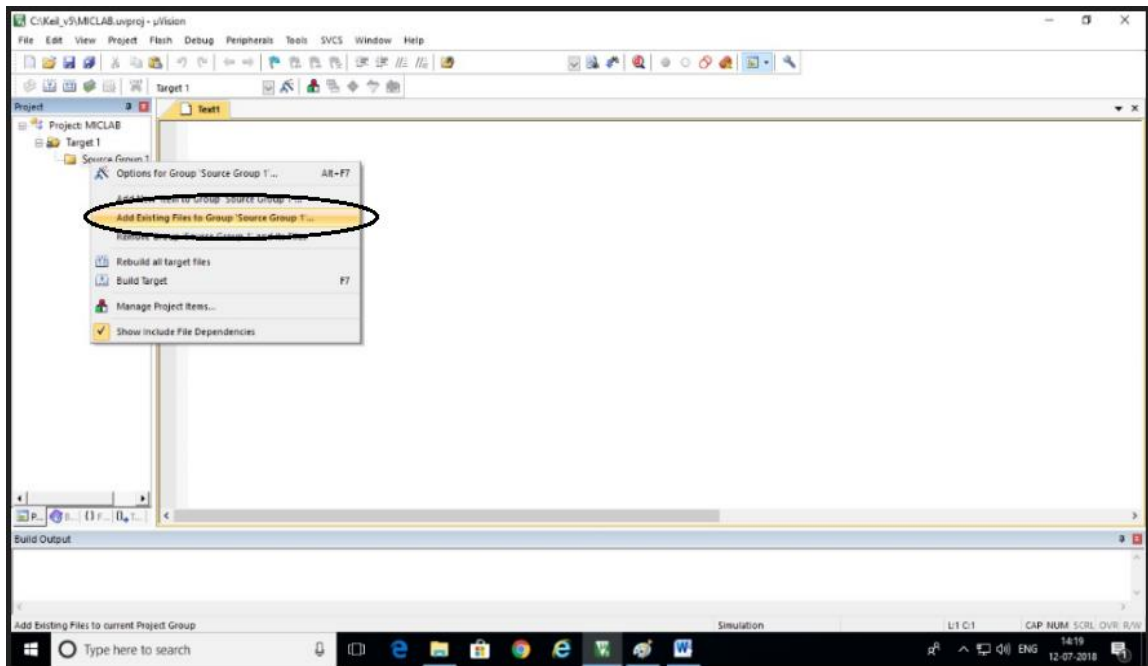
4. "Select device for Target Target-1" window will open. It displays a list of manufacturers of microcontrollers.
5. Double click on ATMEL or INTEL, list of supported microcontrollers gets displayed. Select 80C51AH from INTEL or AT89C51 (or as per the target board) for ATMEL then click ok.



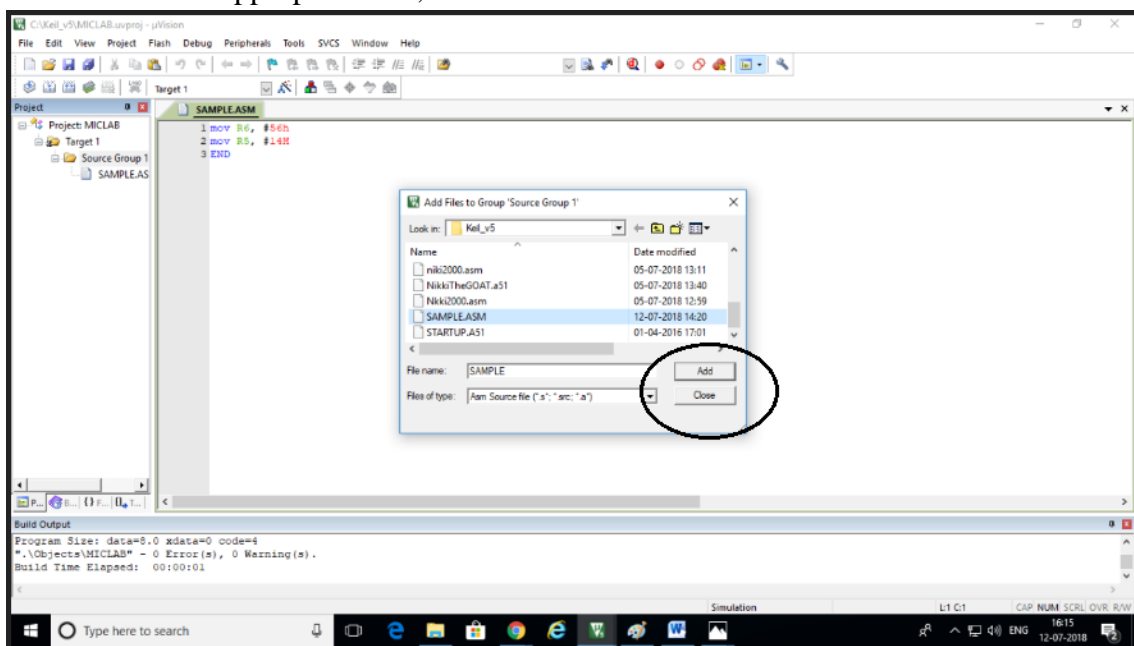
6. Click file pull down menu. Select new, a text editor window will open. Save this file in a same folder where project was stored. Give extension as .ASM or .A51.



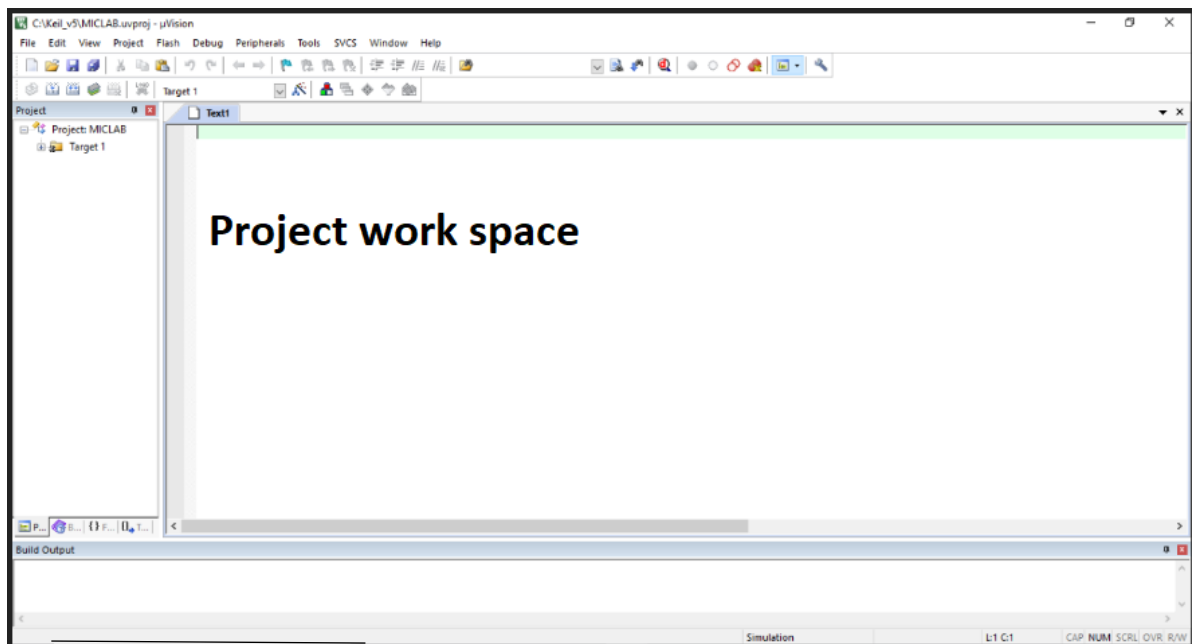
7. On left hand project work space window will display Target1 and Source group1.
8. Right click on source group; Add files to source group 1.



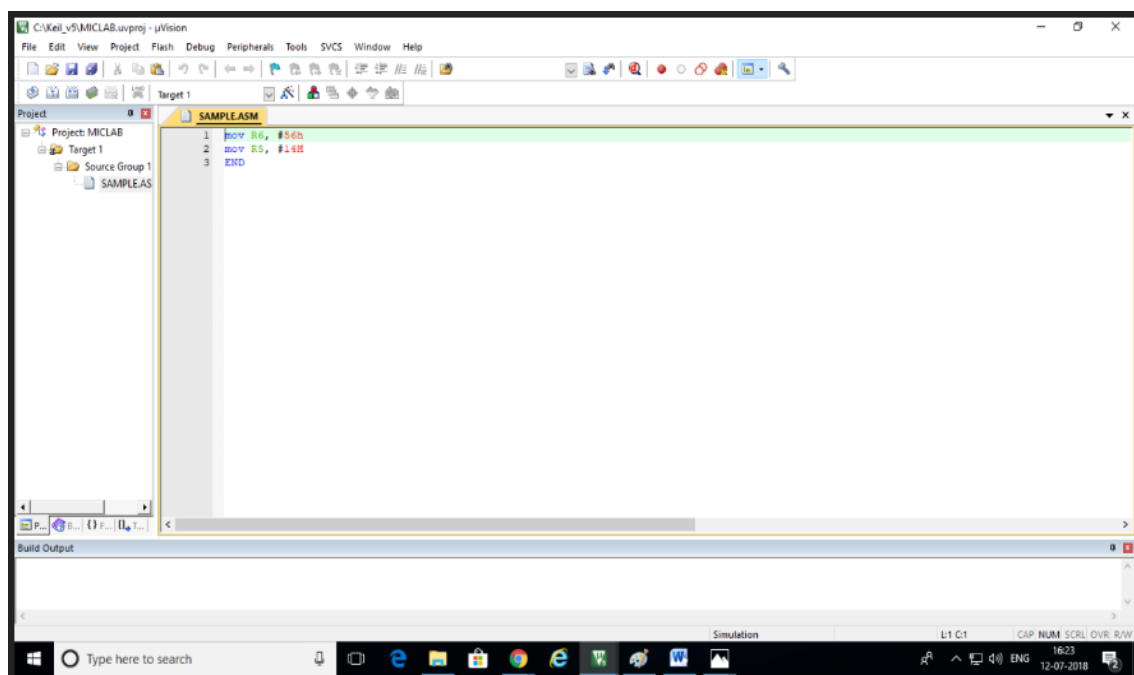
9. Select file type as asm source file. Now all .asm file names will be displayed. Select appropriate file, click ADD and close.



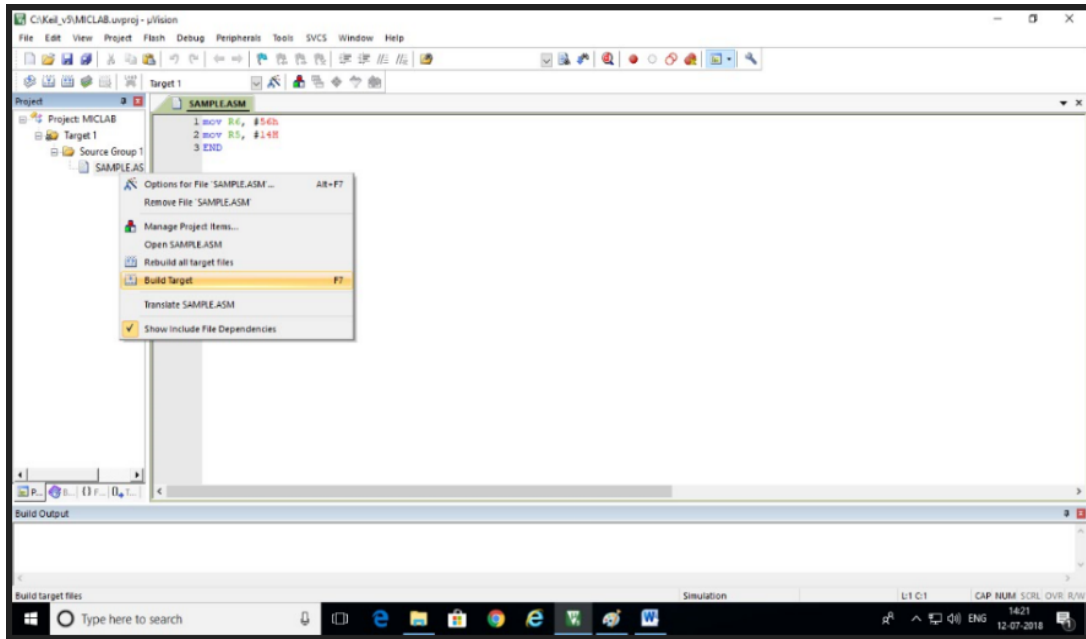
10. Project work space window will display 'Target 1' and 'Source group 1' with added file name.



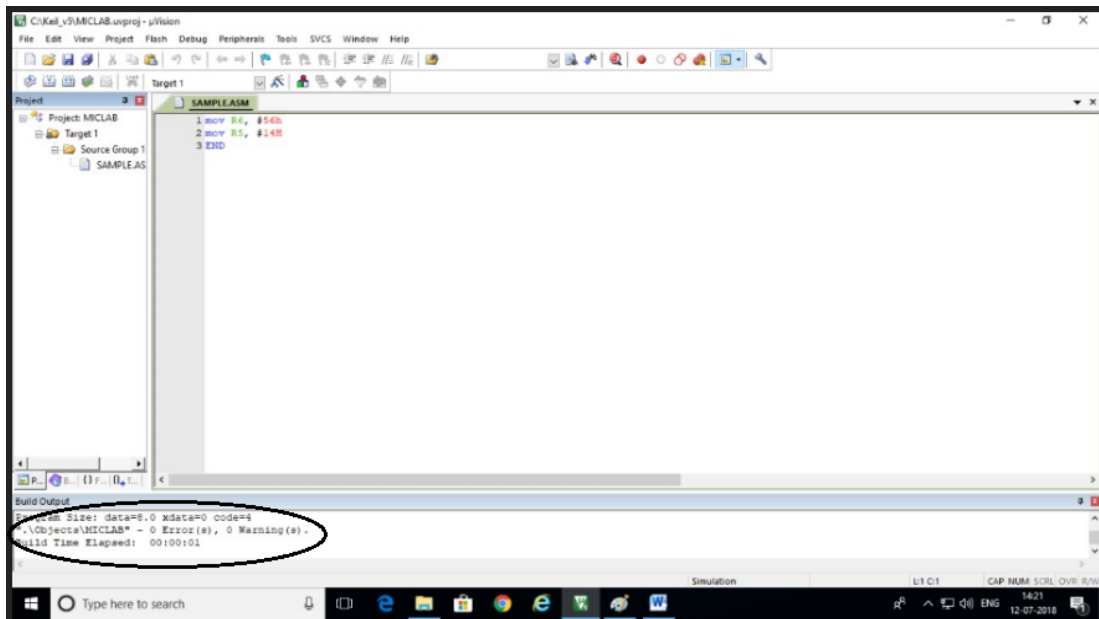
11. Type assembly language program. End with END directive. Save the file periodically.



12. Right click on source group, click on Build target or press F7.



13. Output window will display the errors if any. If there are some errors, then remove the errors and repeat from step number 12 until no errors.

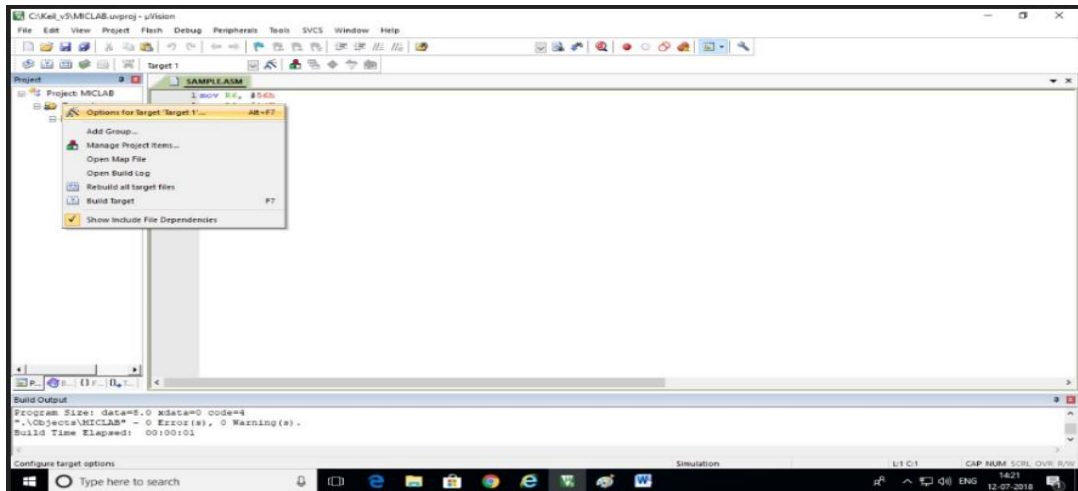


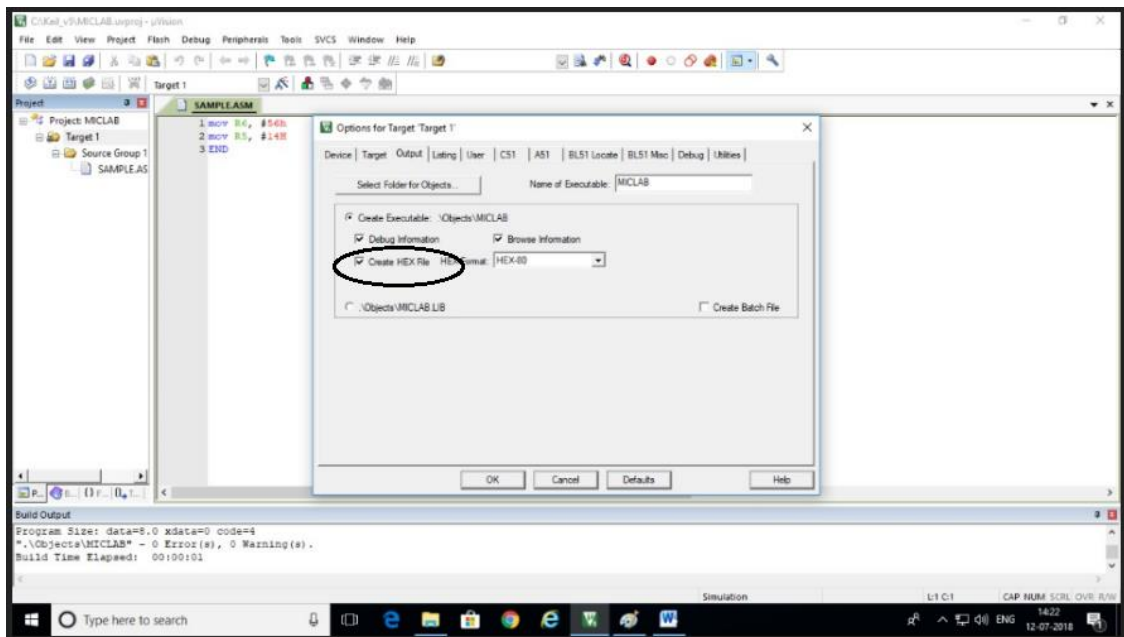
To create a hex file, follow this procedure

- a. Right click on target in project window.
- b. Click on options for target 'target 1'.
- c. Click on output tab and checkmark the option "create hex file".
- d. Click ok
- e. Repeat step 12 again.

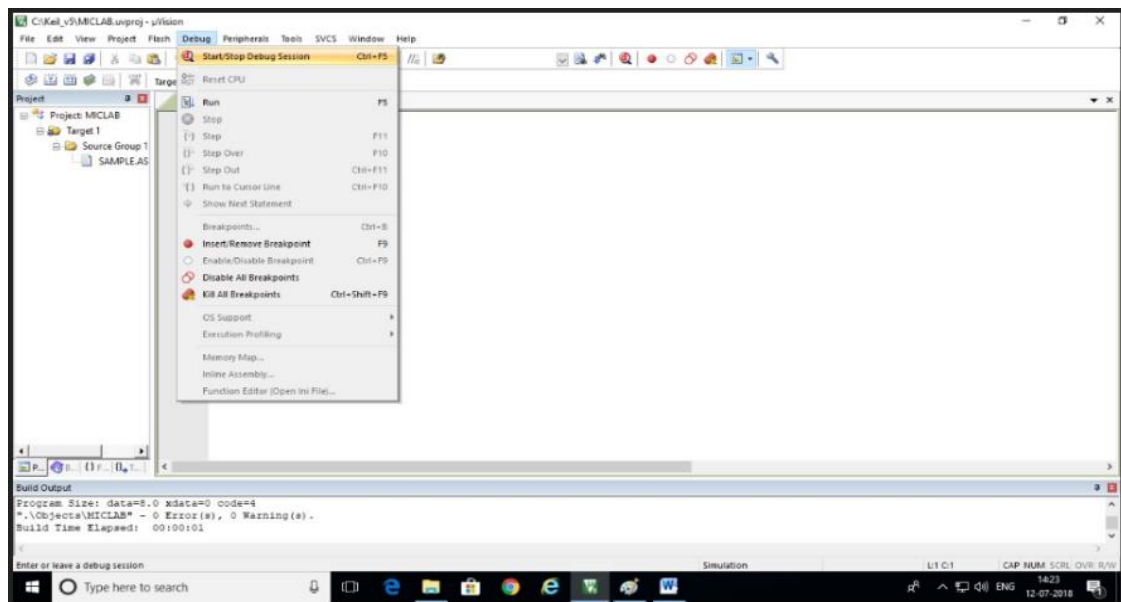
Observe output window. Hex file is created.

This step is optional for the experiments which need only simulation method to observe the results.



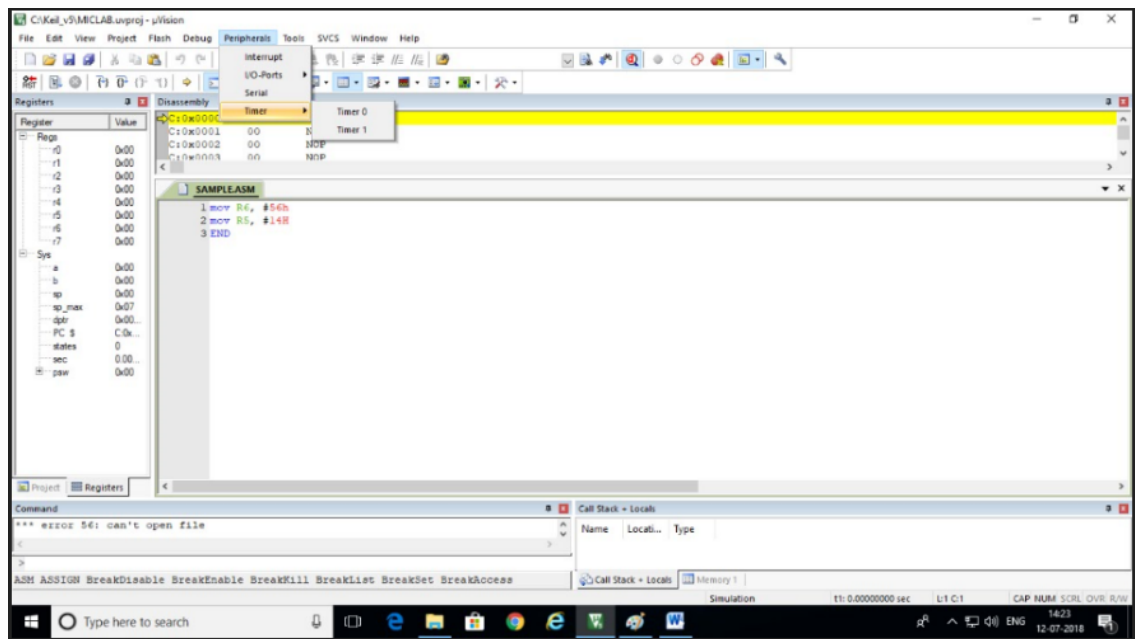


14. To start the simulation. Click on Debug pull down. Then select start/Stop debug session.

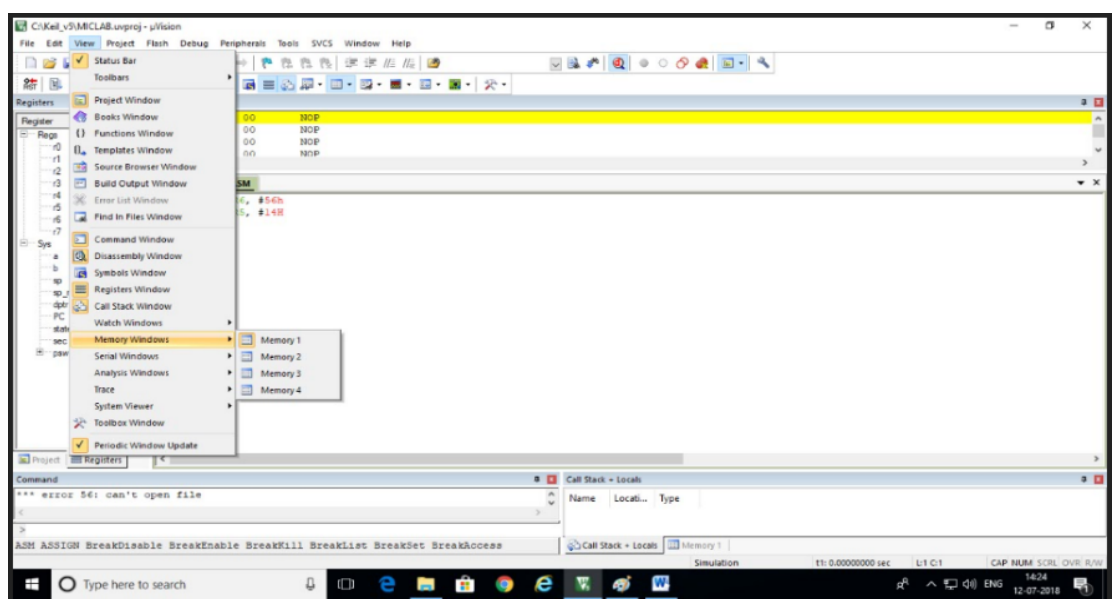


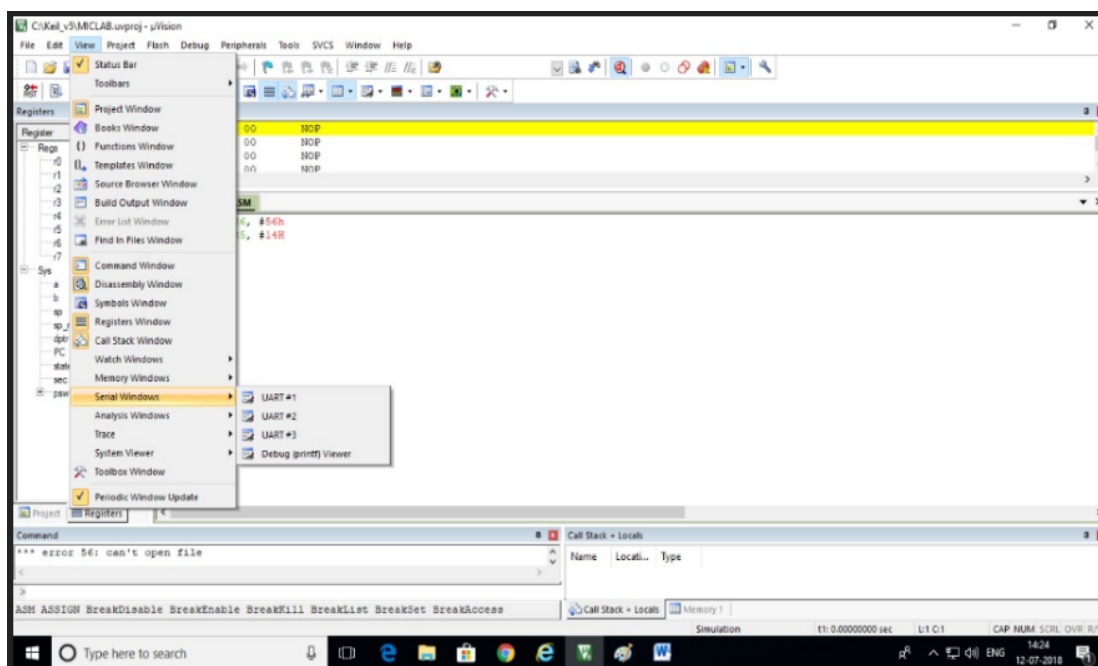
15. On start of debug session, project window will display all internal registers of 8051 and their contents. To execute the program step by step, go on clicking on “step over” button.
16. Observe the logic levels of port pins, timers, interrupt etc. by clicking on PERIPHERALS and select appropriate option.

Execute the program step by step and observe the logic levels on port pins.



17. Observe the serial communication by clicking VIEW pull down and select serial window-1 Option.
18. Observe the contents of internal, external and code memory contents.
 - a. Click on memory window button
 - b. Memory window will get displayed near output window with address bar.
 - c. Type “i: address 8-bit H” for internal memory.” X: address16 bit H” for external memory for code memory type “C: address”.
 - d. To modify the contents of memory, right click on contents of any memory location and enter new valued to be loaded in that memory location





Please note that the features of Sr. no.16 to 18 are available only in debug mode

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

X. Resources Used

S. No.	Instrument /Components	Specification	Quantity
1.			
2.			

XI. Actual procedure followed (use blank sheet provided if space is not sufficient)

1.
2.
3.
4.
5.

XII. Observations

Observe development board and list various components and write their functions.

Table no. 1

Sr. No.	Component	Function

XIII. Result (Identification of various block of 8051 development board)

.....

XIV. Interpretation of Results (Hardware and software features of 8051 development board and Keil)

.....

XV. Conclusions and Recommendations (Actions/decisions to be taken based on the interpretation of results).

.....

XVI. Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO.

1. Define bus. List different types of buses available in 8051 microcontroller
2. List any four features of Keil IDE.
3. State Program memory and Data memory capacity of 8051 microcontroller
4. Explain the selection criteria for selecting a microcontroller device.
5. State function of following pins of 8051 microcontroller:
 i) ALE ii) RST iii) \overline{EA} iv) \overline{PSEN}

XVIII. Assessment Scheme

Performance indicators		Weightage
Process related(15 Marks)		60% (15)
1	Use of IDE tools for programming	30%
2	Identifying components on developer kit	20%
3	Follow ethical practices.	10%
Product related (10 Marks)		40%(10)
4	Observations and recording	20%
5	Relevance of output of the problem definition.	15%
6	Timely Submission of report, Answer to sample questions.	05%
TOTAL		100% (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No.2: Assembly Language Program using various addressing modes

I Practical Significance

The addressing modes specifies the way data can be moved or copied from source to destination location thus providing various options and flexibility for data transfer. This allows the programmer to write structured program which is essential to code maintainability.

II Industry/Employer Expected Outcome(s)

Maintain microcontroller based systems.

III Course Level Learning Outcome(s)

Develop program in 8051 in assembly language for the given operation.

III Laboratory Learning Outcome(s)

Develop an Assembly Language Program (ALP) for addition of two number bers using various addressing modes and assembler directives.

V Relevant Affective Domain related outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

Addressing Modes

A microcontroller provides various methods for accumulator essing data needed in the execution of an instruction. The various methods of accumulator essing the data are called addressing modes.

1. Immediate addressing mode:

The data is provided in instruction itself.

Ex: MOV A, #05H (The immediate data 05H provided in instruction is moved into A register).

Ex: ADD A, #05H (The immediate data 05H provided in instruction is added with contents of A register and the result is stored in A register).

2. Register addressing mode:

The registers hold the data. The permitted registers are A, R7-R0 of each register bank.

Ex: MOV A, R0 content of R0 register is copied into Accumulator.

Ex: ADD A, R0 content of R0 register is added with content of Accumulator and result is stored in Accumulator.

3. Direct addressing mode:

The data is in the RAM memory location and this address is given as part of instruction.

Ex: MOV A, 30H Content of RAM address 30H is copied into Accumulator.

Ex: ADD A, 40H Content of RAM address 40H is added with content of Accumulator and result is stored into Accumulator

4. Register Indirect addressing mode:

Here the address of memory location is indirectly provided by a register.

The '@' sign indicates that the register holds the address of memory location

Ex: MOV A, @R0 Copy the content of memory location whose address is given in R0 register.

Ex: ADD A, @R0 Add content of Accumulator and content of memory location whose address is given by register R0 and store the result in Accumulator.

5. Register specific mode:

The operand is specified by certain specific registers such as accumulator or DPTR

Ex: RRA Rotate the contents of accumulator to the right

6. Indexed Addressing mode:

This addressing mode is basically used for accumulator addressing data from look up table. Here the address of memory is indexed.

Ex: MOVC A, @A+DPTR The content of A register is added with content of DPTR and the resultant is the address of memory location from where the data is copied to A register.

Assembler Directives:

The assembler directives are instructions to the assembler to carry out certain activity during the assembly process. The common assembler directives are:

- ❖ ORG: Indicates the beginning of the address.
- ❖ DB: Used to define 8-bit data in decimal, binary, hexadecimal, ASCII formats.
- ❖ EQU: Used to define a constant without occupying a memory location.
- ❖ END: Indicates end of the source file.

VII Required Resources/apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1.	Desktop PC	Loaded with open-source IDE, simulation and program downloading software.	1 No.

VIII Precautions to be followed

- 1) Check rules / syntax of assembly language programming.

IX Procedure

Develop Program

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL or INTEL and select 80c51AH or AT89C51.
5. Type the program in text editor and save as .asm or .a51.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window. It will display all internal registers of 8051 and their contents.
11. Note the contents of the registers in observation table

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

X Resources Used

S. No.	Instrument /Components	Specification	Quantity
1.			
2.			

SAMPLE PROGRAM 1: Write a program to add two 8-bit numbers using immediate and direct addressing mode.

Step 1: Algorithm

1. Start
2. Move the immediate data 02H into accumulator
3. Add the data in accumulator with immediate data 03H
4. Initialize R1 with memory address 30H
5. Move the result of addition from accumulator to the memory address pointed by R1.
6. Stop

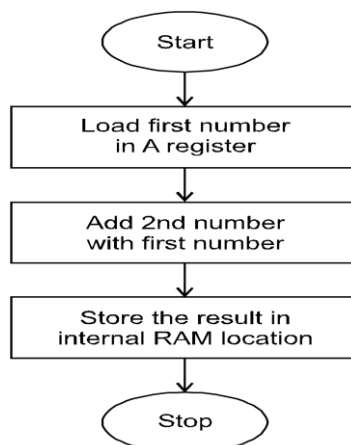
Step 2-Flowchart

Fig 2.1 Flowchart to add data from accumulator with immediate data

Step 3: Assembly Language Program

Memory Address	Hex Code		Label	Mnemonics	Comments
				ORG 0000H	
C:0x0000	7402			MOV A, #02H	;Move the data 02H in accumulator
C:0x0002	2403			ADD A, #03H	;Add the contents of accumulator with 03H
C:0x0004	F530			MOV 30H, A	;Move the result in accumulator to memory address 30H
				END	;Stop

Output Window

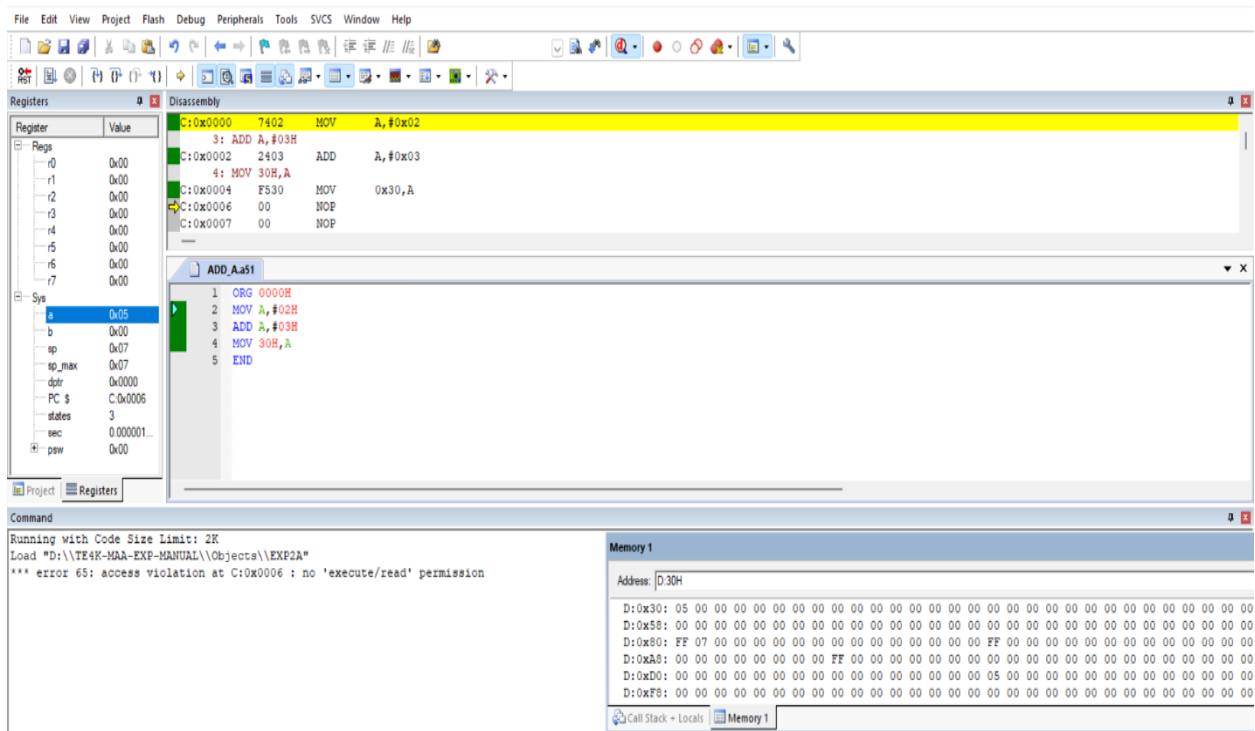


Fig 2.2 Output Window

SAMPLE PROGRAM 2: Write a program using ORG, DB and END directives

Step 1-Algorithm

1. Start.
2. Use ORG directive to set memory location at 0000H.
3. Initialize data pointer to memory location pointed by label MYDATA.

4. Clear Accumulator.
5. Move into accumulator a byte of data located at address pointed by DPTR.
6. Move data in accumulator to register r0
7. Increment data pointer
8. Clear Accumulator.
9. Move into accumulator a byte of data located at address pointed by DPTR.
10. Add data in accumulator with data in register R0
11. Initialize Data pointer to new memory location 2000H.
12. Move the contents of Accumulator to address pointed by DPTR.
13. Use data directive to write data 02H at memory location 0010H and data 03H at memory location 0011H

Step 2-Flowchart

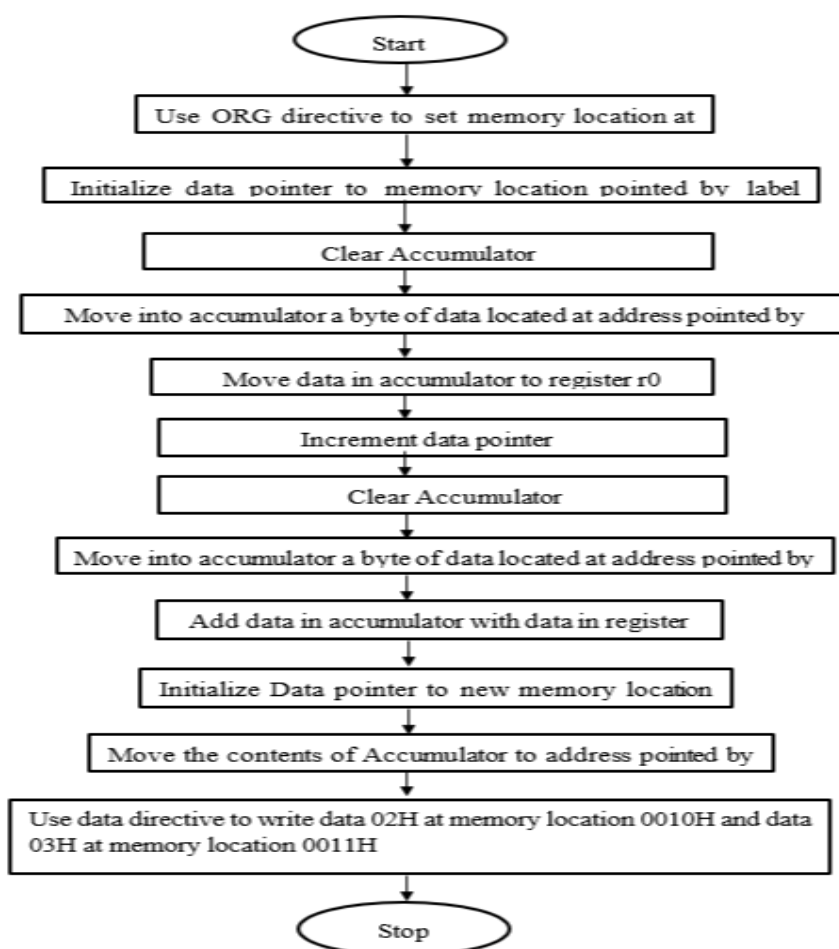


Fig 2.3 Flowchart for ORG, DB and END directives program

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	;Use ORG directive to set memory location at 0000H
0000	900010		MOV DPTR, #MYDATA	;Initialize data pointer to memory Location pointed by label MYDATA
0003	E4		CLR A	; Clear Accumulator
0004	93		MOVC A, @A+DPTR	;Move code byte at ACCUMULATOR +DPTR to ACCUMULATOR
0005	F8		MOV R0, A	;Move data in accumulator to register R0
0006	A3		INC DPTR	;Increment data pointer by 1
0007	E4		CLR A	; Clear Accumulator
0008	93		MOVC A, @A+DPTR	;Move code byte at accumulator +DPTR to accumulator
0009	28		ADD A, R0	;Add data in accumulator with data in Register R0
000A	902000		MOV DPTR, #2000H	;Initialize data pointer to new data memory location 2000H
000D	F0		MOVX @DPTR, A	Move the contents of accumulator to address pointed by DPTR
000E	80FE		SJMP \$	
0010	0203	MYDATA:	DB 02H,03H	;8-bit Data bytes to be added
			END	;Stop

X Resources used

Sr. No.	Name of Resource	Specifications	Quantity

XI Actual Procedure Followed (use blank sheet provided if space not sufficient)

1.
2.
3.
4.
5.
6.
7.
8.
9.

XII Observation Table

After execution of sample program 1

After execution of sample program 2

Accumulator	
Immediate data	
30H memory location	

Accumulator	
R0	
2000H memory location	

XIII Results (Output of the Program)

.....

XIV Interpretation of Results (Give meaning of the above obtained results)

.....

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 3: ALP to perform arithmetic operations on 8-bit data

I Practical Significance

Applications of microcontroller often involve performing mathematical calculations. 8051 microcontroller provide arithmetic instructions for performing operations such as addition, subtraction, multiplication, division etc. This practical will help the students to develop skills to write assembly program for arithmetic operations.

IV Industry/Employer Expected Outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Develop program in 8051 in assembly language for the given operation.

V Laboratory Learning Outcome(s)

Develop an ALP to perform arithmetic operations: addition, subtraction, multiplication and division on 8-bit data

V Relevant Affective Domain related outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

Arithmetic operations of addition, subtraction, multiplication and division are carried out by Register A. Register B is specifically used for multiplication and division purpose. The 8051 microcontroller consists of four register banks, Bank0, Bank1, Bank2, Bank3. Each bank contains 8 registers (R0 to R7). Arithmetic operations affect flags in PSW register of 8051

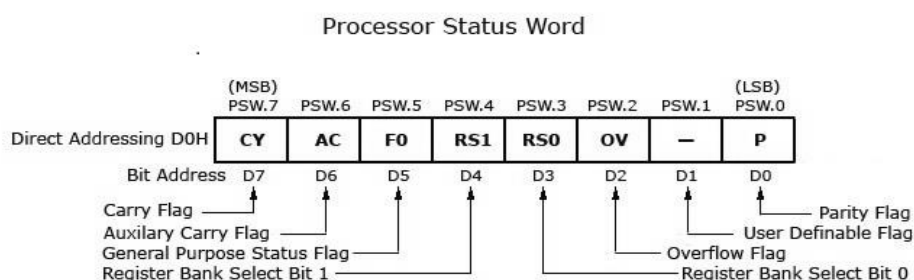


Fig 3.2 Program Status Word Register

Arithmetic Instructions:

Mnemonics	Operational description
ADD A, #number	Add the immediate number with accumulator and stores result in accumulator
ADD A, Rn	Add the data in Rn with accumulator and stores result in accumulator
ADD A, add	Add the data in add with accumulator and stores result in accumulator
ADD A, @Rp	Add the data at the address in Rp with accumulator and stores result in accumulator
ADDC A, # number	Add the immediate number with accumulator and carry, stores result in accumulator
ADDC A, Rn	Add the data in Rn with accumulator and carry, stores result in accumulator
ADDC A, add	Add the data in add with accumulator and carry, stores result in accumulator
ADDC A, @Rp	Add the data at the address in Rp with accumulator and carry, stores result in accumulator
SUBB A, #number	Subtract immediate number and carry from accumulator; stores the result in accumulator
SUBB A, add	Subtract the content of add and carry from accumulator; stores the result in accumulator
SUBB A, Rn	Subtract the data in Rn and carry from accumulator; stores the result in accumulator
SUBB A, @Rp	Subtract the data at the address in Rp and carry from accumulator; stores the result in accumulator
MUL AB	Multiply accumulator and register B. store the lower byte of result in accumulator and higher byte in B
DIV AB	Divide accumulator by register B. store quotient in accumulator and remainder in B
INC A	Increments the accumulator by 1
INC Rn	Increments the data in register Rn by 1

INC @Rp	Increments the data at the address in Rp
DEC A	Decrements the accumulator by 1
DEC Rn	Decrements the data in register Rn by 1
DEC @Rp	Decrements the data at the address in Rp
INC DPTR	Increments data pointer by 1

XII Required Resources/apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1.	Desktop PC	Loaded with open-source IDE, simulation and program downloading software.	1 No.

XIII Precautions to be followed

- 1) Check rules / syntax of assembly language programming.

XIV Procedure

Develop Program

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL or INTEL and select 80c51AH or AT89C51.
5. Type the program in text editor and save as .asm or .a51.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window. It will display all internal registers of 8051 and their contents.
11. Note the contents of the registers in observation table

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write a program to add, subtract, multiply and divide two 8 bit numbers.

Step 1: Algorithm

1. Move data 04H in Accumulator.
2. Add data 03H to the number stored in accumulator (04 H).
3. Store result of addition operation in register R1
4. Clear Carry flag
5. Move data 08H in accumulator.
6. Subtract data 02H from number stored in accumulator (08 H).
7. Store result of subtraction operation in register R2
8. Move data 05H in accumulator and data 03H in register B.
9. Multiply the two number.
10. Store lower byte of product in R3 and higher byte of product R4.
11. Move data 10 H in Accumulator and 06 H in register B.
12. Divide the two number.
13. Store quotient in register R5 and remainder in register R6
14. Stop

Step 2-Flowchart

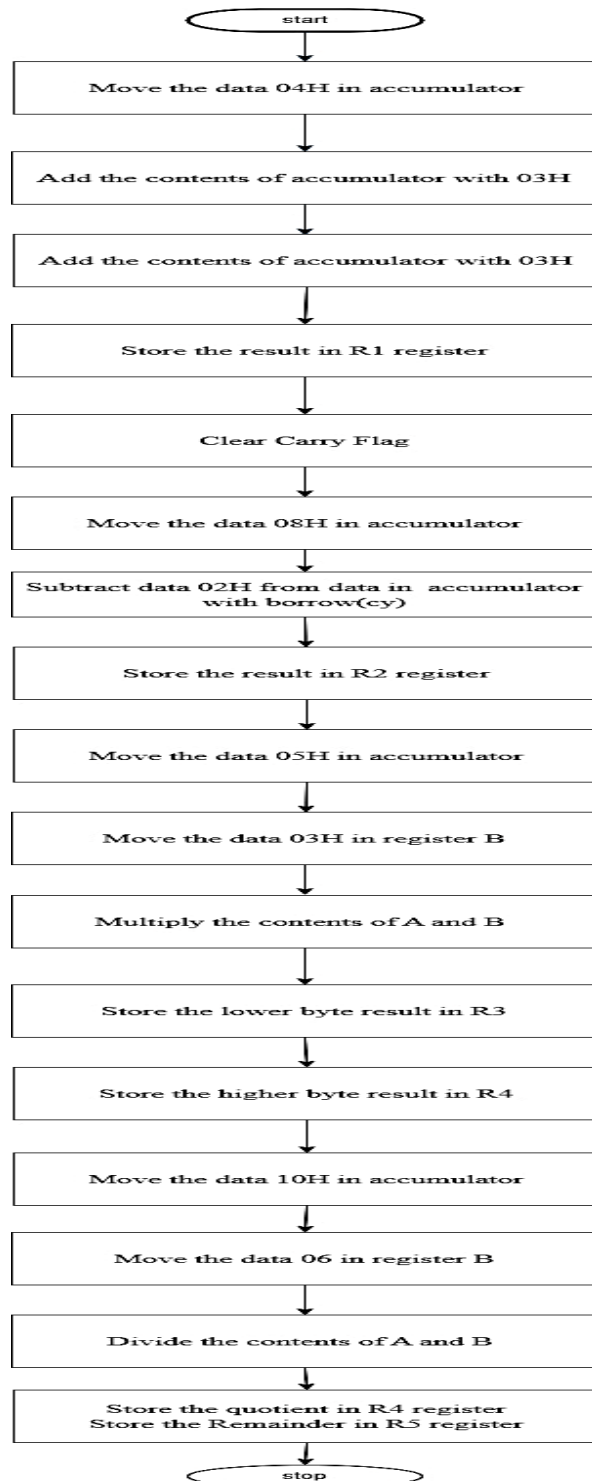
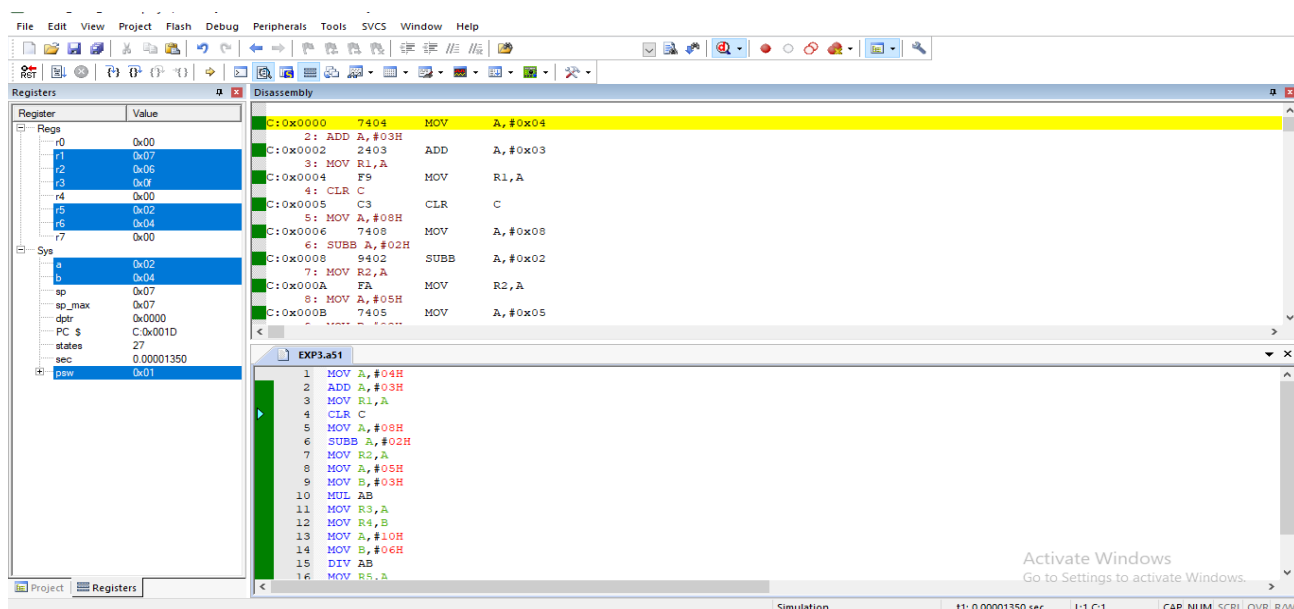


Fig 3.3 Flowchart for arithmetic operations

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0x0000	7404		MOV A, #04H	; Move the data 04H in accumulator
C:0x0002	2403		ADD A, #03H	; Add the contents of accumulator with 03H
C:0x0004	F9		MOV R1, A	; Store the result in R1 register
C:0x0005	C3		CLR C	;Clear the carry flag
C:0x0006	7408		MOV A, #08H	;Move the data 08H in accumulator
C:0x0008	9402		SUBB A, #02H	;Subtract the 02H from data in accumulator with borrow(cy)
C:0x000A	FA		MOV R2, A	;Store the result in R2 register
C:0x000B	7405		MOV A, #05H	;Move the data 05H in accumulator
C:0x000D	75F003		MOV B, #03H	;Move the data 03H in register B
C:0x0010	A4		MUL AB	;Multiply the contents of A and B
C:0x0011	FB		MOV R3, A	;Store the lower byte result in R3
C:0x0012	ACF0		MOV R4, B	;Store the higher byte result in R4
C:0x0014	7410		MOV A, #10H	;Move the data 10H in accumulator
C:0x0016	75F006		MOV B, #06H	;Move the data 06 in register B
C:0x0019	84		DIV AB	;Divide the contents of A and B
C:0x001A	FD		MOV R4, A	;Store the quotient in R4 register
C:0x001B	AEF0		MOV R5, B	;Store the Remainder in R5 register
			END	;Stop

Output Window**Fig 3.4 Output Window**

XV Resources used

Sr. No.	Name of Resource	Specifications	Quantity

XVI Actual Procedure Followed (use blank sheet provided if space not sufficient)

1.
2.
3.
4.
5.
6.
7.
8.

XII Observations for sample program (use blank sheet provided if space not sufficient)

Arithmetic operation	DATA BYTE1	DATA BYTE 2	Result after execution
Addition			R1 =
Subtraction			R2 =
Multiplication			R3 = R4 =
Division			R5 = R6 =

XIII Results (Output of the Program)

.....

XIV Interpretation of Results (Give meaning of the above obtained results)

.....

.....

.....

.....

.....

.....

.....

XVII References/Suggestions for further reading

1. <https://www.tutorialspoint.com/arithmetric-group-in-8051>
2. <https://technobyte.org/arithmetric-instructions-8051/>
3. <https://technobyte.org/arithmetric-instructions-8051/>

XVIII Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 4: ALP to perform arithmetic operations on 16-bit data.

I Practical Significance

8051 microcontrollers have single instruction arithmetic operations. Applications such as BCD and ASCII conversions and checksum byte testing require arithmetic operations. This practical will help the students to develop skills to write assembly program for arithmetic operations.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Develop program in 8051 in assembly language for the given operation.

IV Laboratory Learning Outcome(s)

Develop an ALP to perform arithmetic operations: addition, subtraction on 16-bit data.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

When performing 16-bit addition on an 8051 microcontroller, you typically use a combination of registers to store the intermediate values, operands, and results. The registers used for 16-bit arithmetic operations are:

1. Accumulator (A): Used for arithmetic and logical operations.
2. Data Pointer (DPTR): A 16-bit register used to point to memory locations. Often used to point to the locations of the operands in memory.
3. General-Purpose Registers (R0 - R7): Used to store data temporarily.

Register bank and their RAM address

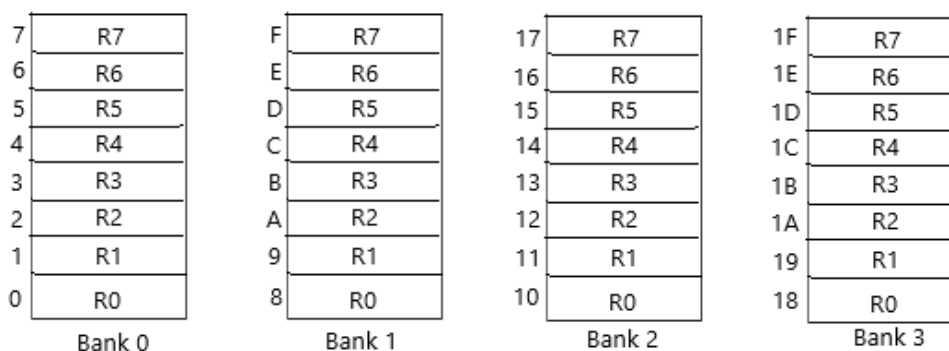


Fig 4.1 Register Banks

8051 uses DPTR, a 16 bit register to access the 16-bit data from external memory. It is used in MOVX, MOVC command

Arithmetic operations affect flags in PSW register of 8051

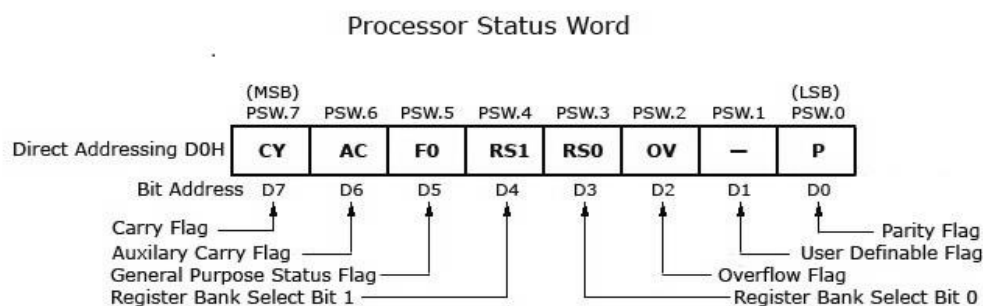


Fig 4.2 Program Status Word Register

The 8051 microcontroller supports various arithmetic operations using specific instructions. Here's an overview of the key instructions used for arithmetic operations:

Addition Instructions

ADD A, source: Adds the source operand to the accumulator (A). Source can be a register (R0-R7), a direct address, or an immediate value.

Example: ADD A, R1: Adds the value in register R1 to the accumulator

ADDC A, source: Adds the source operand to the accumulator along with the carry bit. Used for multi-byte (e.g., 16-bit) addition where carry needs to be considered.

Example: ADDC A, R2: Adds the value in register R2 and the carry bit to the accumulator

Subtraction Instructions

SUBB A, source: Subtracts the source operand and the carry bit from the accumulator. Source can be a register, a direct address, or an immediate value.

Example: SUBB A, #10H: Subtracts the immediate value 10H and the carry bit from the accumulator

VII Resources Required

Sr. No.	Instrument /Components	Specification	Quantity
1	Desktop PC	Loaded with open-source IDE, simulation and program downloading software	1 No.

VIII Precautions to be Followed

1. Check rules / syntax of assembly programming.

IX Procedure

Write Program

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

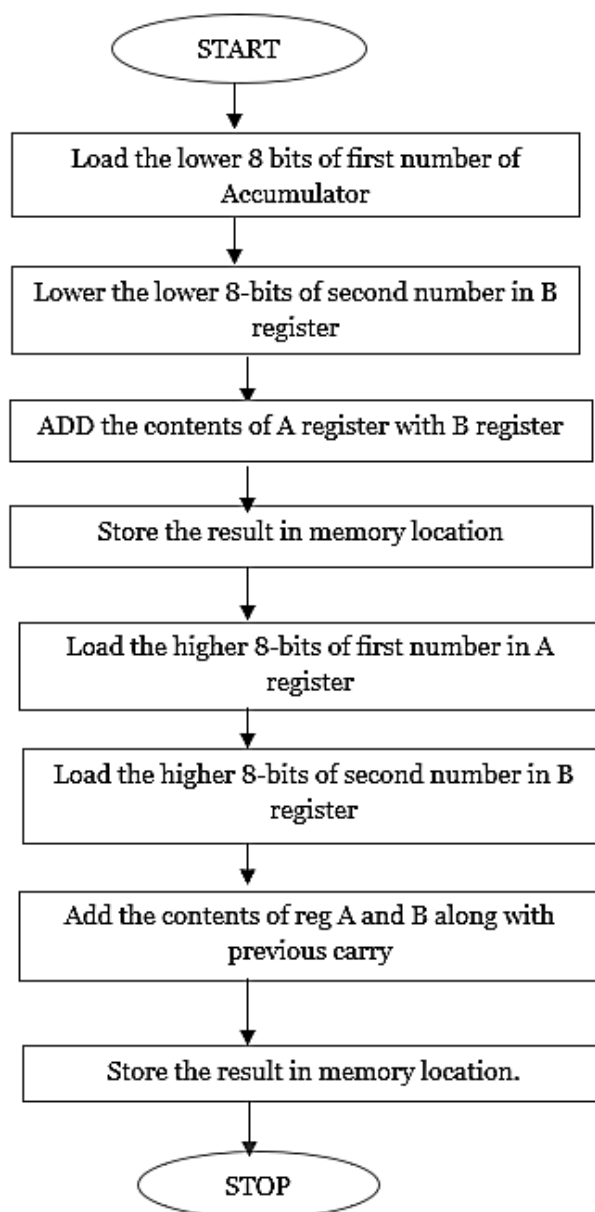
E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: To perform following: arithmetic operations: 16-bit addition

Step 1-Algorithm

1. Select register bank 2.
2. Initialize Carry Counter to get result > 16 bit.
3. Load LSB of first number in Accumulator.
4. Add LSB of Second number with LSB of first number.
5. Store LSB of result.
6. Load MSB of first number in Accumulator.
7. Add MSB of second number with MSB of first Number.
8. If Carry is not 1, then go to step 10.
9. Increment Carry Counter by 1.
10. Store MSB of result.
11. Store Carry of result.
12. Stop.

Step 2-Flow Chart**Fig 4.3 Flowchart for arithmetic operations****Step 3- Assembly Language Sample Program: 16-bit Addition**

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0x0000	E540		MOV A, 40H	Load the contents of memory location 40H in A register (Lower 8-bits of first number)

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x0002	8542F0		MOV B, 42H	Load the contents of memory location 42H in B register (Lower 8 bits of second number)
C:0x0005	25F0		ADD A, B	Add the contents of A and B
C:0x0007	F544		MOV 44H, A	Store the result in memory location 44H
C:0x0009	E541		MOV A, 41H	Load the content of 41H in register A (Higher 8 bits of first number)
C:0x000B	8543F0		MOV B, 43H	Load the contents of 43H in register B (Higher 8 bits of second number)
C:0x000E	35F0		ADDC A,B	ADD the contents of A and B with previous carry
C:0x0010	F545		MOV 45H, A	Store the result at location 45H
			END	

Output Window

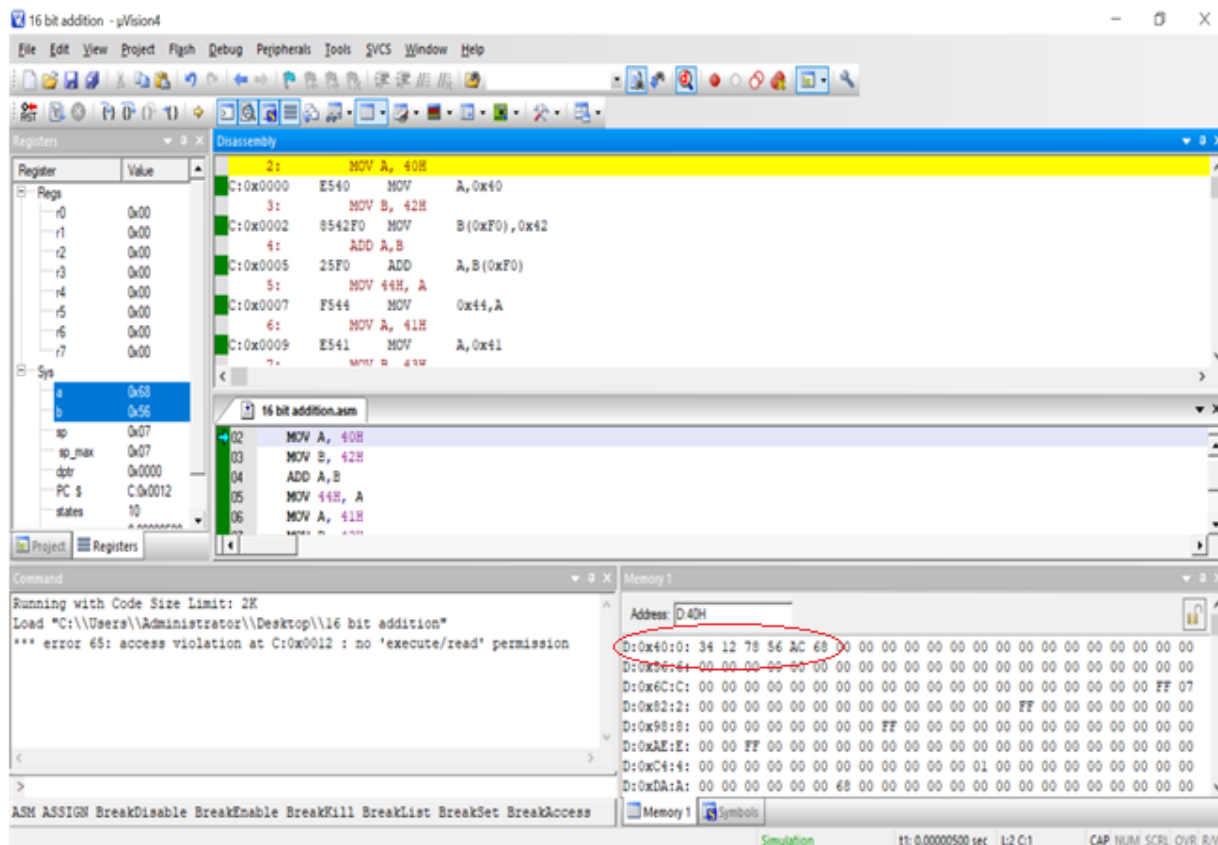


Fig 4.4 Output Window

Problem statement 1 for student: Write a program to perform 16-bit subtraction

Step 1-Algorithm	Step 2-Flowchart
-------------------------	-------------------------

Step 3- Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments

Memory Address	Hex Code	Label	Mnemonics	Comments

X Resources Used

S. No.	Instrument /Components	Specification	Quantity

XI Actual Procedure Followed (use blank sheet provided if space not sufficient)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XII Observations for sample program (use blank sheet provided if space not sufficient)

Sr. No.	Registers/Memory Locations used in the code	Contents / Result after execution
1		
2		
3		

Sr. No.	Registers/Memory Locations used in the code	Contents / Result after execution
4		
5		
6		

XIII Results (Output of the Program)

.....

.....

.....

.....

XIV Interpretation of Results (Give meaning of the above obtained results)

.....

.....

.....

.....

XV Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....

.....

.....

.....

XVI Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. Write the status of the CY, AC and P flag after the addition of 1239CH and AC64H.
2. Write instructions to perform subtraction without borrow.
3. Write instructions to perform the following operations:
 - a. Set the carry flag.
 - b. Select Bank 2 of RAM memory.

[Space for Answers]

.....

.....

XVIII Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 5: ALP to perform addition of BCD data.

I Practical Significance

8051 microcontrollers have single instruction arithmetic operations. Applications such as BCD and ASCII conversions and checksum byte testing require arithmetic operations. This practical will help the students to develop skills to write assembly program for arithmetic operations.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Develop program in 8051 in assembly language for the given operation.

IV Laboratory Learning Outcome(s)

Develop an ALP to perform addition of BCD data stored at external and store result in internal memory.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

Binary-Coded Decimal (BCD) is a method of representing decimal numbers where each digit is encoded as a separate 4-bit binary number. This allows for easier manipulation and display of decimal numbers in digital systems, such as microcontrollers and computers. The numbers from 0 to 9 are valid BCD whereas the numbers from A to F are invalid BCDs. Binary-Coded Decimal (BCD) addition in the 8051 microcontroller involves adding two BCD numbers and adjusting the result to ensure it remains in BCD format. If the result of addition results in an invalid BCD then suitable modification of addition of six is done to convert the number from invalid to valid BCD. For this in 8051 DAA instruction is used.

DAA Instruction:

The DAA (Decimal Adjust Accumulator) instruction is used in the 8051 microcontrollers to correct the result of a binary-coded decimal (BCD) addition operation. After adding two BCD numbers, the result might not be a valid BCD number. The DAA instruction adjusts the accumulator to ensure the result is a valid BCD number.

Conditions for Adjustment:

1. If the lower 4 bits (nibble) of the accumulator are greater than 9, or if the auxiliary carry (AC) flag is set, 6 is added to the lower nibble.
2. If the upper 4 bits (nibble) of the accumulator are greater than 9, or if the carry (C) flag is set, 6 is added to the upper nibble.

Ex: If the result of the addition is 0x3C (which is not a valid BCD), DA A will correct it to 0x42 (which is a valid BCD for the number 42).

VII Resources Required

Sr. No.	Instrument /Components	Specification	Quantity
1	Desktop PC	Loaded with open-source IDE, simulation and program downloading software	1 No.

VIII Precautions to be Followed

1. Check rules / syntax of assembly programming.

IX Procedure

Write Program

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: To perform addition of BCD data stored at external and store the result in internal memory.

Step 1-Algorithm

1. Initialize Data pointer [DPTR] with memory location 2000H
2. Load the contents of memory location pointed by DPTR to Accumulator
3. Transfer the contents of A register to R0.
4. Increment DPTR
5. Load the contents of memory location pointed by DPTR to Accumulator
6. Perform the addition of contents of A register with R0 register.

7. Perform BCD adjustment on the result by using DA A instruction.
8. Move the contents of Accumulator (result) to internal memory location
9. Stop.

Step 2-Flow Chart:

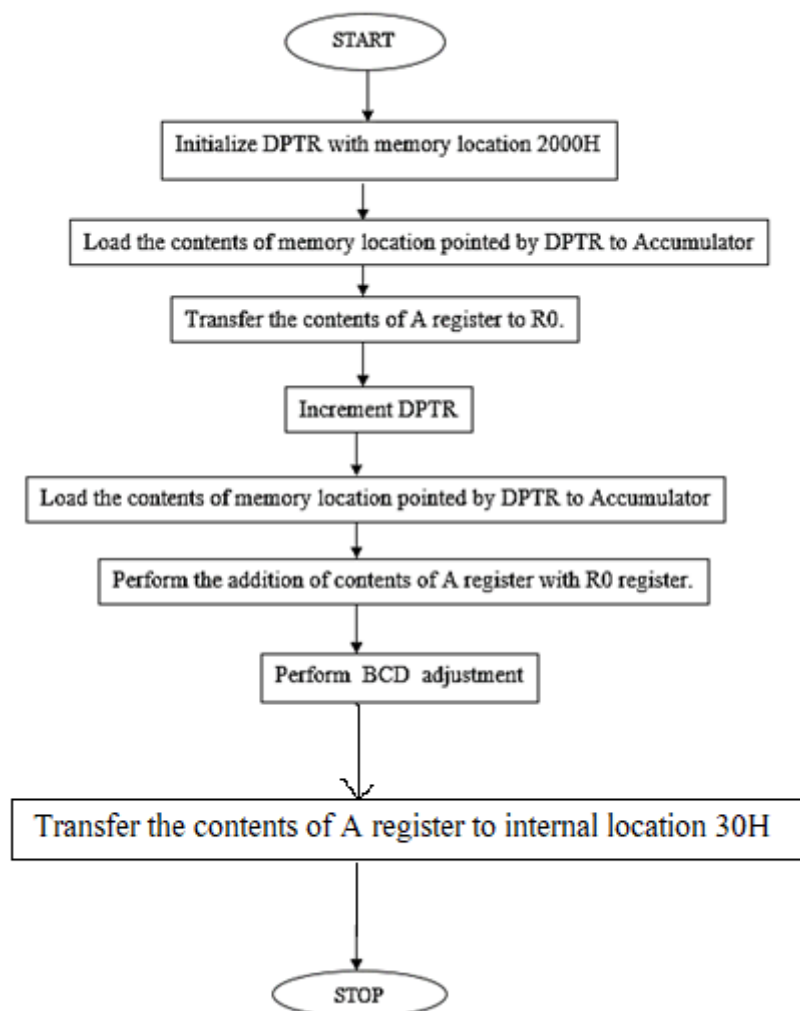


Fig 5.1 Flowchart for BCD addition

Step 3- Assembly Language Sample Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 00H	
C:0x0000	902000		MOV DPTR, #2000H	Initialize Data pointer [DPTR] with memory location 2000H
C:0x0003	E0		MOVX A, @DPTR	Load the contents of memory location pointed by DPTR to Accumulator
C:0x0004	F8		MOV R0, A	Transfer the contents of A register to R0.
C:0x0005	A3		INC DPTR	Increment DPTR

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x0006	E0		MOVX A ,@DPTR	Load the contents of memory location pointed by DPTR to Accumulator
C:0x0007	28		ADD A, R0	Perform the addition of contents of A register with R0 register
C:0x0008	D4		DA A	Perform BCD adjustment on the result by using DA A instruction.
C:0x0009	F530		MOV 30H, A	Transfer the result to internal memory location 30H.
			END	

Input and Output Window:

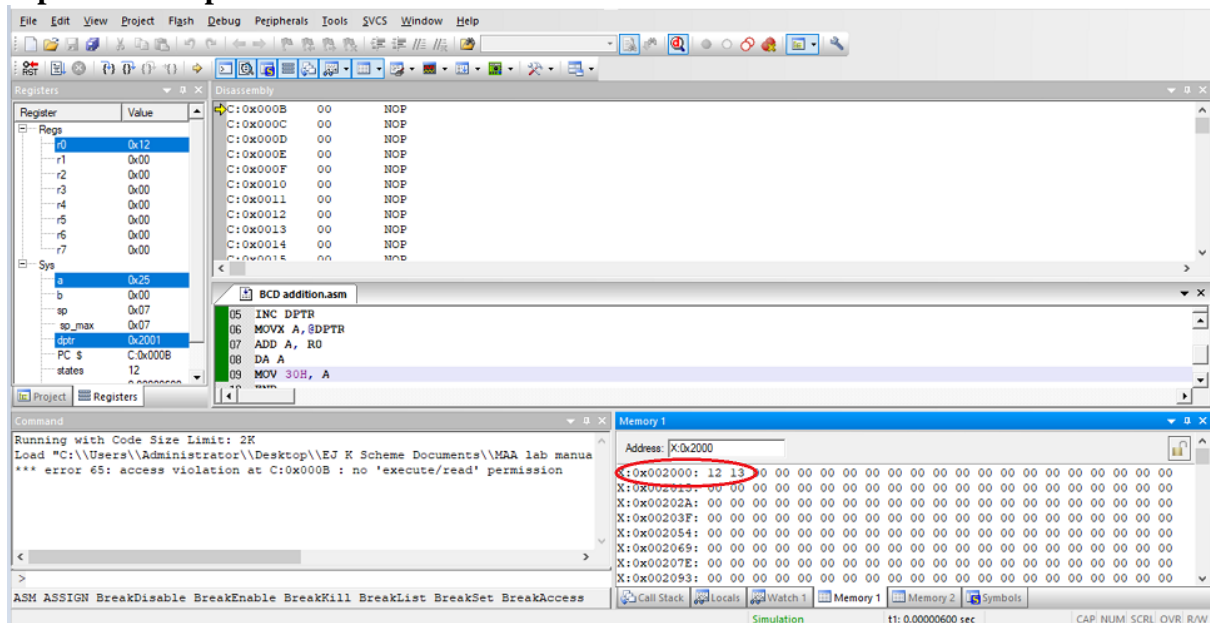


Fig 5.2 Input Window

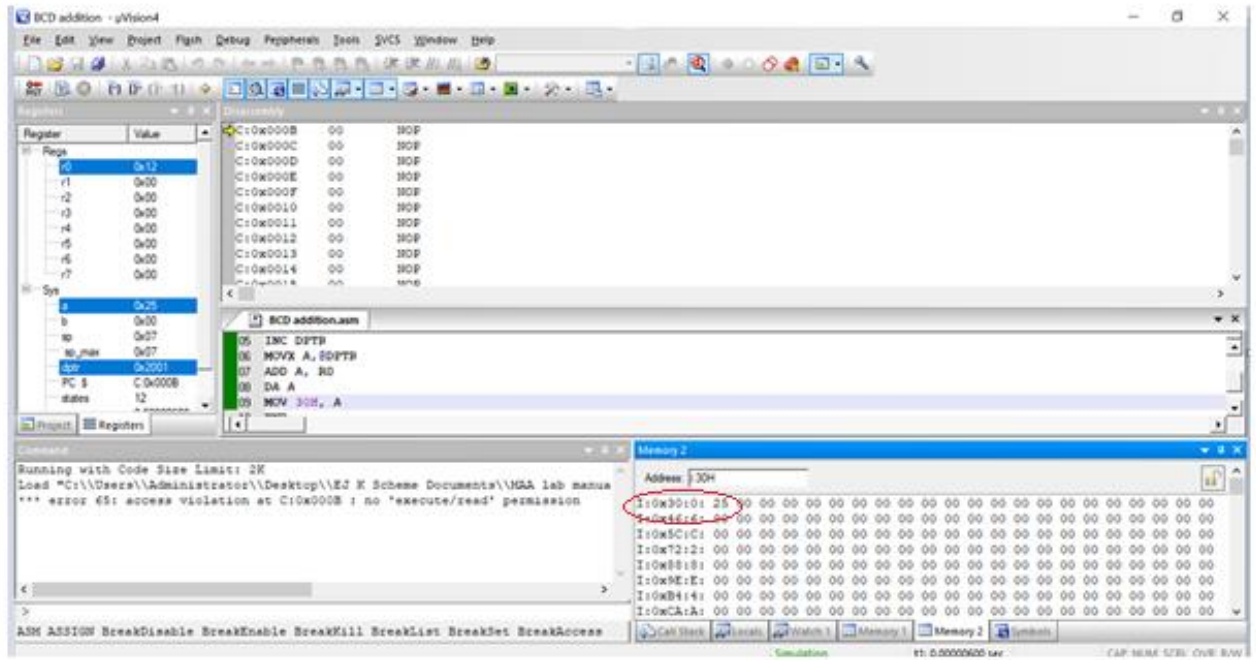


Fig 5.2 Output Window

Problem statement 1 for student: Write a program to perform BCD subtraction

<p>Step 1-Algorithm</p>	<p>Step 2-Flowchart</p>
--------------------------------	--------------------------------

Step 3- Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments

X Resources Used:

S. No.	Instrument /Components	Specification	Quantity

XI Actual Procedure Followed (use blank sheet provided if space not sufficient)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XIII Precautions Followed (use blank sheet provided if space not sufficient)

.....

.....

.....
.....
.....
.....

XIV Observations for sample program (use blank sheet provided if space not sufficient)

Sr. No.	Memory Location used in the code	Contents after execution
1		
2		
3		

XV Results (Output of the Program)

.....
.....
.....
.....

XVI Interpretation of Results (Give meaning of the above obtained results)

.....
.....
.....
.....

XVII Conclusions and Recommendation (Actions/decisions to be taken based on the Interpretation of results).

.....
.....
.....
.....

XIX References / Suggestions for further reading

1. <https://www.refreshnotes.com/2016/04/8051-program-addition-8bit-2digit-bcd.html>
2. <https://www.tutorialspoint.com/binary-coded-decimal-bcd-addition>
3. <https://www.vlsifacts.com/bcd-addition/>

XX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 6: ALP to perform series addition.

I Practical Significance

The series addition in the 8051 microcontroller is a fundamental aspect of its programming and functionality. Series addition is a basic arithmetic operation that is used extensively in various programming tasks within the microcontroller. It is often employed in loop mechanisms where iterative addition of values is required, essential for tasks like averaging or cumulative sums.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Develop program in 8051 in assembly language for the given operation.

IV Laboratory Learning Outcome(s)

Develop an ALP for sum of series stored in RAM locations 40 to 49H. Find the sum of the values at the end of program the lower byte stored in 30H and the high byte in 31H.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

The 8051 microcontrollers for performing series addition uses INC and DEC instructions for efficient byte increment and decrement operations, along with loop instructions and several iterations to get the final result.

INC (Increment Instruction): The INC instruction increases the value of a byte by one.

Syntax: INC operand

Operands: Can be an accumulator (A), a register (R0-R7), a direct address, a data pointer (DPTR), or an indirectly addressed memory location (@Ri).

Example: INC A: Increments the accumulator and result stored in accumulator.

DEC (Decrement Instruction): The DEC instruction decreases the value of a byte by one.

Syntax: DEC operand

Operands: Similar to INC, it can target the accumulator, a register, a direct address, or an indirectly addressed memory location.

Example: DEC A: Decrements the accumulator and result stored in accumulator.

Branching Instructions:

Operation	Mnemonics	Description
Call	ACALL Address11	Calls a subroutine in the maximum address range of 2K bytes
	LCALL Address16	Calls a subroutine in the maximum address range of 64K bytes
Return	RET	Returns the control from subroutine
	RETI	Returns the control from an interrupt subroutine
Jump	AJMP Address11	Jumps to an address in a 2KB range
	LJMP Address16	Jumps to an address in a 64KB range
	SJMP Relative address	Jumps to an address in a 256-byte range (0 to 127 (0-7FH) range and -1 to -128 (FFH-80H)).
	JMP @A+DPTR	[DPTR]<-[DPTR+A]
	JZ Relative address	Jumps to address when accumulator=0
	JNZ Relative address	Jumps to address when accumulator ≠0
	CJNE A, Direct address, Relative address	Jumps to relative address when accumulator=data stored at a direct address
	CJNE A, #Data, Relative address	Jumps to relative address when accumulator=data given by the programmer
	CJNE @Rn, #Data, Relative address	Jumps to relative address when data at memory location stored in register=data given by the programmer
	DJNZ Rn, Relative address	Decrements value in Rn and jump to relative address till Rn =0
	DJNZ Direct address, Relative address	Decrements value at memory location stored in a register and jump to relative address till memory location stored in register =0

VII Resources Required

Sr. No.	Instrument /Components	Specification	Quantity
1	Desktop PC	Loaded with open-source IDE, simulation and program downloading software	1 No.

VIII Precautions to be Followed

1. Check rules / syntax of assembly programming.

IX Procedure**Write Program**

1. Start Keil by double clicking on Keil icon.

2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Develop an ALP for sum of series stored in RAM locations 40 to 49H. Find the sum of the values at the end of program the lower byte stored in 30H and the high byte in 31H.

Step 1-Algorithm

1. Initialize Register R0 with the starting memory address.
2. Initialize Register R1 as counter to count the number of data values.
3. Initially clear register R2 and Accumulator.
4. Add the contents of Register A and contents of memory location pointed by R0.
5. Check whether carry flag is set or not. If carry flag is set then Increment register R2
6. Increment Register R0 to point to next number.
7. Decrement register R1 to check whether all additions are performed or not
8. If zero flag is not set then go to step 4 and repeat the process till all numbers are added
9. Move the contents of Accumulator (result) and Register R2 to memory location 30H and 31H.
10. Stop.

Step 2-Flow Chart:

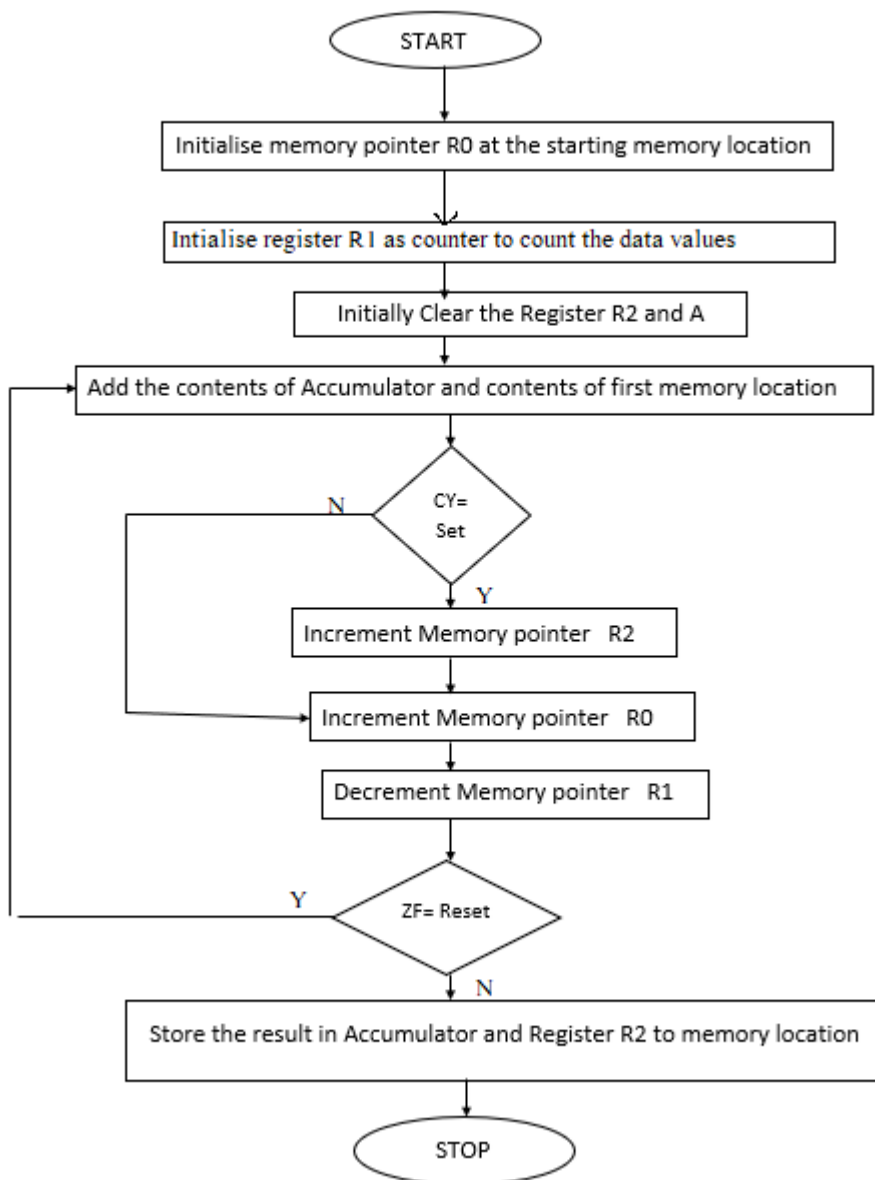


Fig 6.1 Flowchart for Series addition

Step 3- Assembly Language Sample Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 00H	
C:0x0000	7840		MOV R0, #40H	Initialize Register R0 with memory location 40H
C:0x0002	790A		MOV R1, #0AH	Initialize Register R1 with count value of 10 [0AH]
C:0x0004	7A00		MOV R2, #00H	Clear Register R2
C:0x0006	E4		CLR A	Clear Accumulator

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x0007	26	RPT	ADD A, @R0	Add the contents of register A and the contents of memory location pointed by R0.
C:0x0008	5001		JNC DOWN	Check the carry flag if it is not set then jump to label down
C:0x000A	0A		INC R2	Increment the contents of Register R2
C:0x000B	08	DOWN	INC R0	Increment the contents of Register R0
C:0x000C	D9F9		DJNZ R1, RPT	Decrement R1 and check if the contents are zero or non-zero.
C:0X000E	F530		MOV 30H, A	Move the contents of accumulator to memory location 30H
C:0X0010	8A31		MOV 31H, R2	Move the contents of Register R2 to memory location 31H
			END	

Input and Output Window:

Input

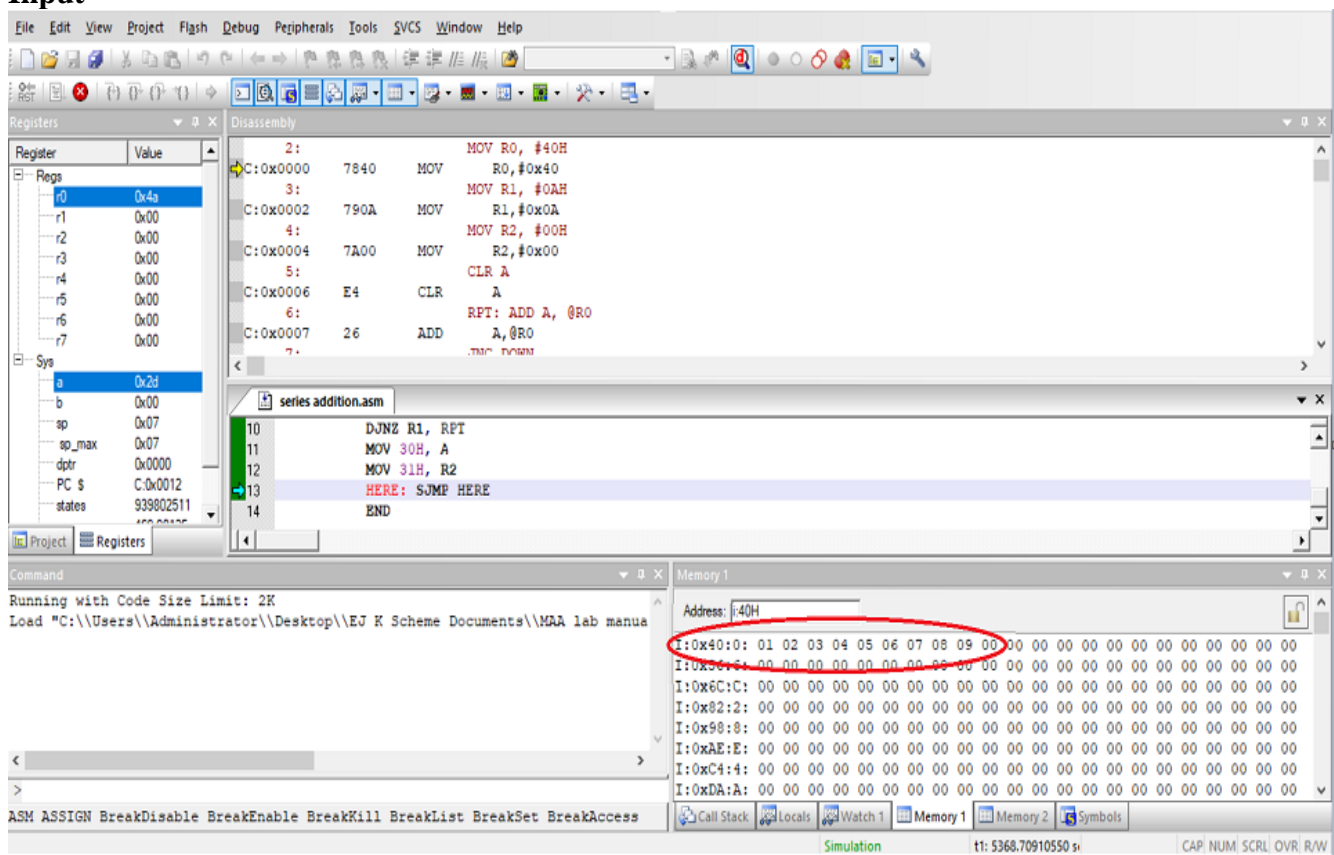


Fig 6.2 Input Window

Output Window

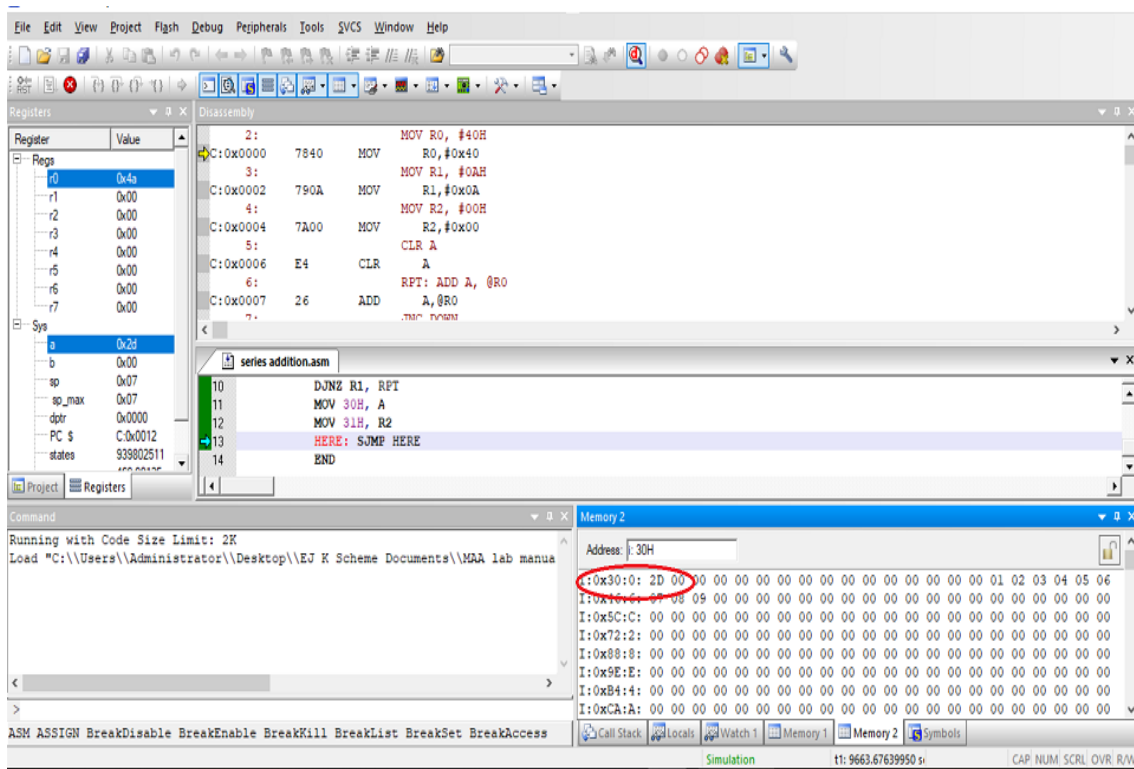


Fig 6.3 Output Window

Problem statement 1 for student: Write a program to perform Series addition of five numbers taken from external memory locations and store the result in registers R0 and R1 respectively.

Step 1-Algorithm	Step 2-Flowchart

.....
.....
XI Observations for sample program (use blank sheet provided if space not sufficient)

Sr. No.	Memory Location used in the code	Contents
1		
2		
3		
4		
5		
6		
7		
8		
9		
10		

Sr. No.	Memory Location used in the code	Contents of execution
1		
2		

XV Results (Output of the Program)

.....
.....
.....
.....

XVI Interpretation of Results (Give meaning of the above obtained results)

.....
.....
.....
.....

XVII Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....

.....

.....

.....

XVIII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. MOV A, # 55H
RLC A
Give the Contents of accumulator and status of carry flag after execution of the above two instructions.
2. Give the instructions used to set and clear the carry flag.
3. Give the sequence of instructions used to perform subtraction without borrow.

[Space for Answers]

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XIX References / Suggestions for further reading

1. <https://electronicsforu.in/8051-program-to-add-an-array-of-numbers/>
2. <https://www.tutorialspoint.com/program-branch-group-in-8051>
3. <https://www.refreshnotes.com/2016/04/8051-program-sum-of-set-of-numbers-in.html>

XX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40% (10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 % (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 7: Array data transfer from source locations to destination locations

I Practical Significance

Understanding 8051 microcontroller memory organization helps in making optimal use of internal RAM and ROM. For applications which require additional memory, the external memory can be accessed. This practical will help the students to develop skills to transfer data from source to destination location.

VI Industry/Employer Expected Outcome(s)

Maintain microcontroller based systems.

III Course Level Learning Outcome(s)

Develop program in 8051 in assembly language for the given operation.

IV Laboratory Learning Outcome(s)

Develop an ALP to transfer data from source to destination locations of internal/ external data memory.

V Relevant Affective Domain related outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

Microcontroller 8051 has two types of memory, **Program Memory** and **Data Memory**. Program Memory (ROM) is used to permanently save the program being executed, while Data Memory (RAM) is used for temporarily storing data and intermediate results created and used during the operation of the microcontroller.

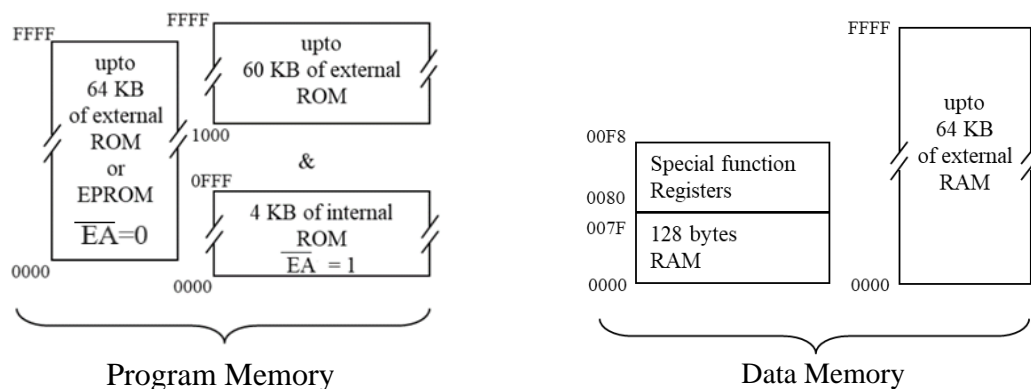


Fig. 7.1 8051 memory organization

The 8051 microcontroller uses data transfer instructions to move data between internal RAM locations, as well as between internal and external RAM.

Internal RAM data transfer instructions:

Table 7.1 : Data Transfer Instruction

Mnemonics	Description
MOV A, direct	Move direct byte to Accumulator
MOV A, @Ri	Move indirect RAM to Accumulator
MOV Rn, direct	Move direct byte to Register
MOV direct, A	Move Accumulator to direct byte
MOV direct, Rn	Move register to direct byte
MOV direct, direct	Move direct byte to direct byte
PUSH direct	Push direct byte onto stack
POP direct	Pop direct byte from stack
XCH A, direct	Exchange direct byte with Accumulator
XCH A, @Ri	Exchange indirect RAM with Accumulator
XCHD A, @Ri	Exchange low-order nibble indirect RAM with Accumulator

Instructions to Access External Data Memory:

Table 7.2: External Data Access Instructions

Mnemonic	Description
MOVX A, @Rp	Copy the contents of the external memory address in Rp to A.
MOVX A, @DPTR	Copy the contents of the external memory address in DPTR to A.
MOVX @Rp, A	Copy data from A to the external memory address in Rp
MOVX @DPTR, A	Copy data from A to the external memory address in DPTR.

XVII Required Resources/apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1.	Desktop PC	Loaded with open-source IDE, simulation and program downloading software.	1 No.

XVIII Precautions to be followed

- 1) Check rules / syntax of assembly language programming.

XIX Procedure**Develop Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL or INTEL and select 80c51AH or AT89C51.
5. Type the program in text editor and save as .asm or .a51.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window. It will display all internal registers of 8051 and their contents.
11. Observe the contents of internal and external data memory.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write a program to transfer a block of 5 data bytes from internal source data memory locations 40H onwards to destination memory locations 50H onwards.

Step 1: Algorithm

1. Set program starting address.
2. Initialize source memory pointer R0 to 40H.
3. Initialize destination memory pointer R1 to 50H.
4. Initialize iteration count R2 to 05H.
5. Move content of first location into accumulator.

6. Move the content of accumulator to first destination location.
7. Increment source memory pointer.
8. Increment destination memory pointer.
9. Decrement iteration count, and jump to step 5, if not zero.
10. Stop

Step 2-Flowchart

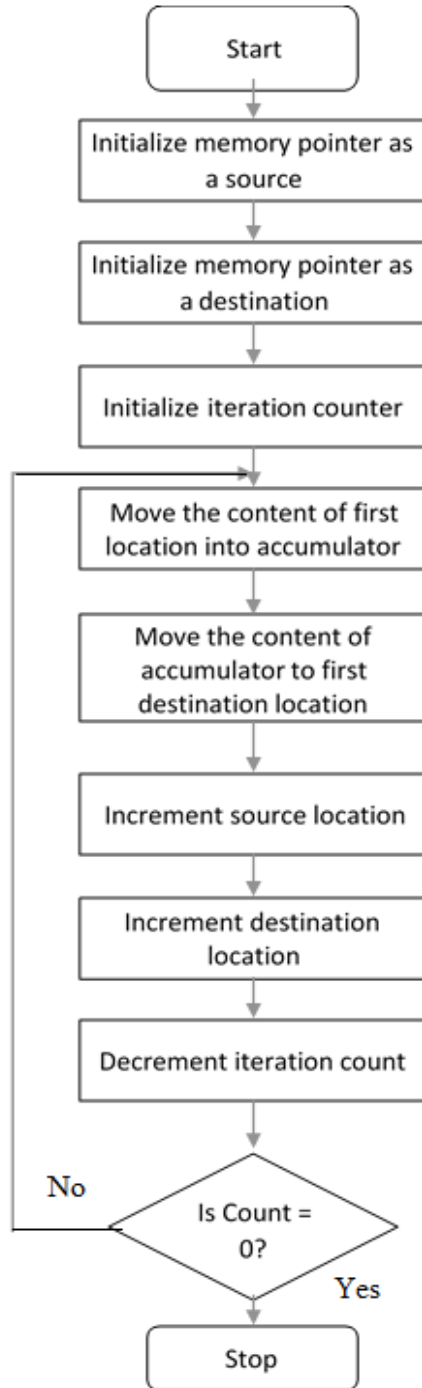


Fig. 7.2 Flowchart to transfer a block of data

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x0000			ORG 0000H	
C:0x0000	7840		MOV R0, #40H	;Initialize source memory pointer R0 to 40H
C:0x0002	7950		MOV R1, #50H	;Initialize destination memory pointer R1 to 50H
C:0x0004	7A05		MOV R2, #05H	;Initialize iteration count to 05H
C:0x0006	E6	UP:	MOV A, @R0	;Move the contents of source memory pointed by R0 to Accumulator
C:0x0007	F7		MOV @R1, A	;Move the contents of Accumulator to destination memory pointed by R1
C:0x0008	08		INC R0	;Increment the contents of R0
C:0x0009	09		INC R1	;Increment the contents of R1
C:0x000A	DAFA		DJNZ R2, UP	; Decrement counter by one, Is it zero? No ,jump to UP
			END	

Output Window

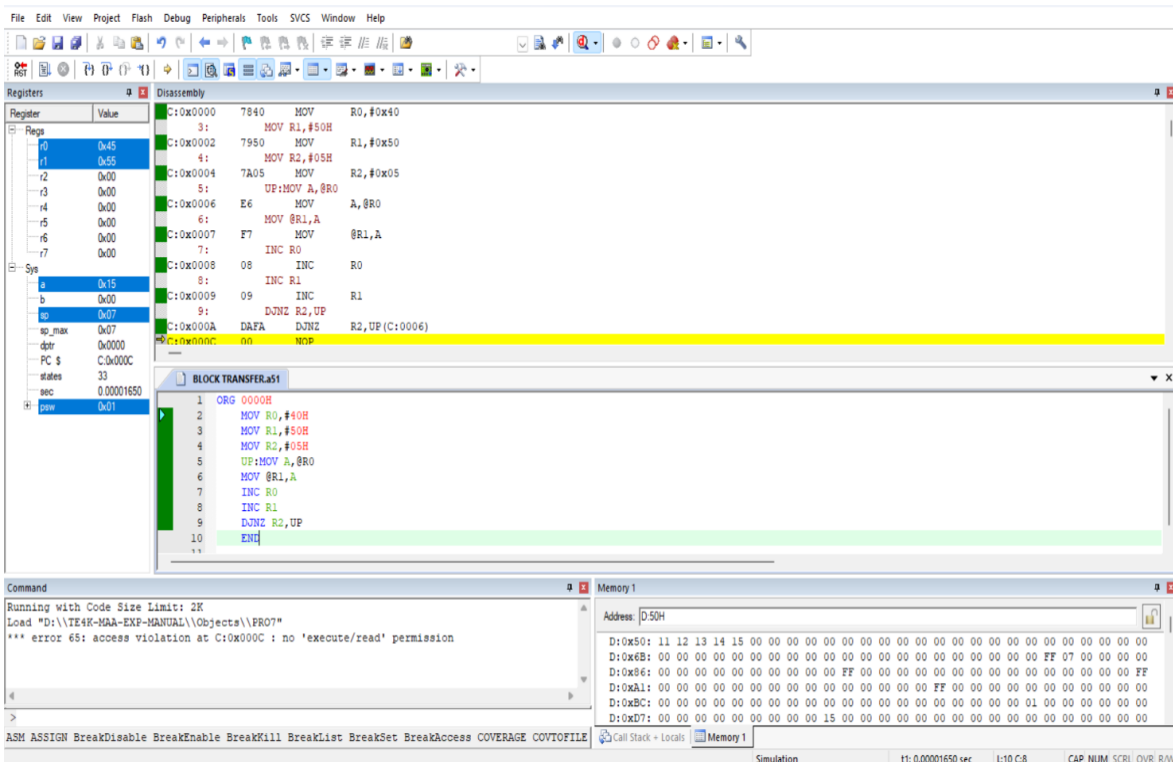


Fig 7.3 Output Window

Problem statement for student: Write a program to transfer a block of 05 bytes from internal data memory location 20H onwards to external data memory location 2000H onwards

Step 1-Algorithm	Step 2-Flowchart

Step 3- Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments

XX Resources used

Sr. No.	Name of Resource	Specifications	Quantity

XXI Actual Procedure Followed (use blank sheet provided if space not sufficient)

1.
2.
3.
4.
5.
6.
7.
8.

XII Observations for problem statement 1 (use blank sheet provided if space not sufficient)

Before execution		After execution	
Memory location	Data	Memory location	Data

XIII Results (Output of the Program)

.....
.....
.....

XIV Interpretation of Results (Give meaning of the above obtained results)

.....
.....
.....

XV Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....
.....

XVI Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifies CO.

1. State any two instructions used to access external memory in 8051.
2. State registers used as memory pointers in 8051.
3. Explain the operation of following instructions:
a) XCHD A, @R1 b) PUSH 30H

[Space for Answers]

.....
.....
.....

.....

.....

.....

.....

.....

.....

.....

XVII References/Suggestions for further reading

1. https://www.silabs.com/documents/public/presentations/8051_Instruction_Set.pdf
2. <https://www.daenotes.com/electronics/digital-electronics/8051-microcontroller-instruction-types>
3. <https://www.electronicshub.org/8051-microcontroller-memory-organization/>

XVIII Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 8: Block exchange of data from source locations to destination locations.

I Practical Significance

Data transfer is a process of moving or copying information from one location to other location within internal or external data memory. To save the results of certain operations, to create lookup tables etc. these data transfer programs are required. Block data transfer is more efficient than byte-by-byte operations, especially when dealing with large data sets. It minimizes overhead by reducing the number of instruction cycles required for data movement.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Develop program in 8051 in assembly language for the given operation.

IV Laboratory Learning Outcome(s)

Develop an ALP to exchange data from source to destination locations of internal/external memory locations.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

The 8051 microcontroller memory is divided into Program Memory and Data Memory. Program Memory (ROM) is used for permanent saving program being executed, while Data Memory (RAM) is used for temporarily storing and keeping intermediate results and variables.

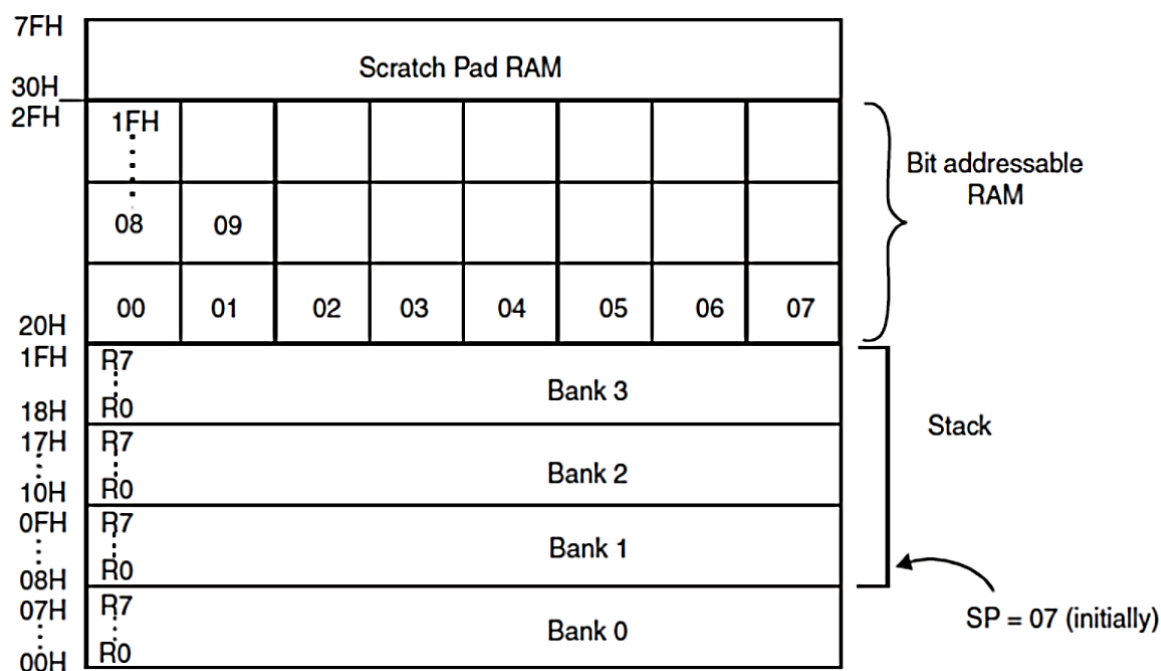


Fig 8.1 RAM Organization in 8051

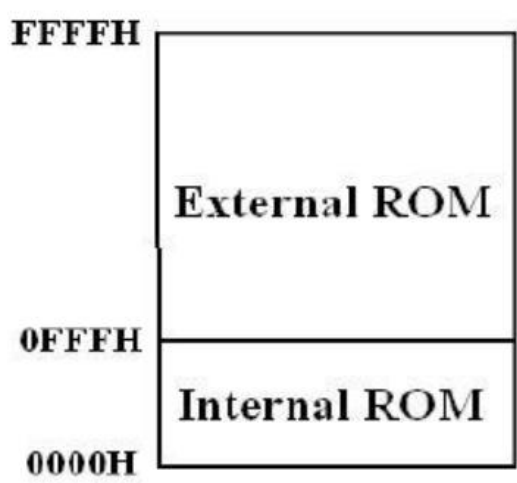


Fig 8.2 ROM Organization in 8051

The 8051 microcontroller includes 128 bytes of internal RAM and up to 4 KB of internal ROM:

1. **RAM (128 bytes):** Divided into working registers, bit-addressable area, and general storage.
2. **ROM (4 KB):** Stores the microcontroller's firmware, typically used for storing the program that the 8051 executes. Locations from 0000H to 0FFFH are internal locations and that exceeds 0FFFH are external locations.

These memory components are integral for the operation and flexibility of the 8051 in various applications.

VII Resources Required

Sr. No.	Instrument /Components	Specification	Quantity
1	Desktop PC	Loaded with open-source IDE, simulation and program downloading software	1 No.

VIII Precautions to be Followed

1. Check rules / syntax of assembly programming.

IX Procedure**Write Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write and execute a program to exchange five data bytes of internal memory location from 50H to 60H onwards.**Step 1-Algorithm**

1. Start
2. Initialize memory pointer as a source.
3. Initialize memory pointer as a destination.
4. Initialize counter.
5. Move the content of first location into accumulator.
6. Move the content of accumulator to first destination location.
7. Increment source location.
8. Increment destination location.
9. Move the contents of Accumulator to source locations

10. If zero flag is not set then go to step 4 and repeat the process till all numbers are added
11. Decrement iteration count and if not zero jump to step 5.
12. Stop.

Step 2-Flow Chart:

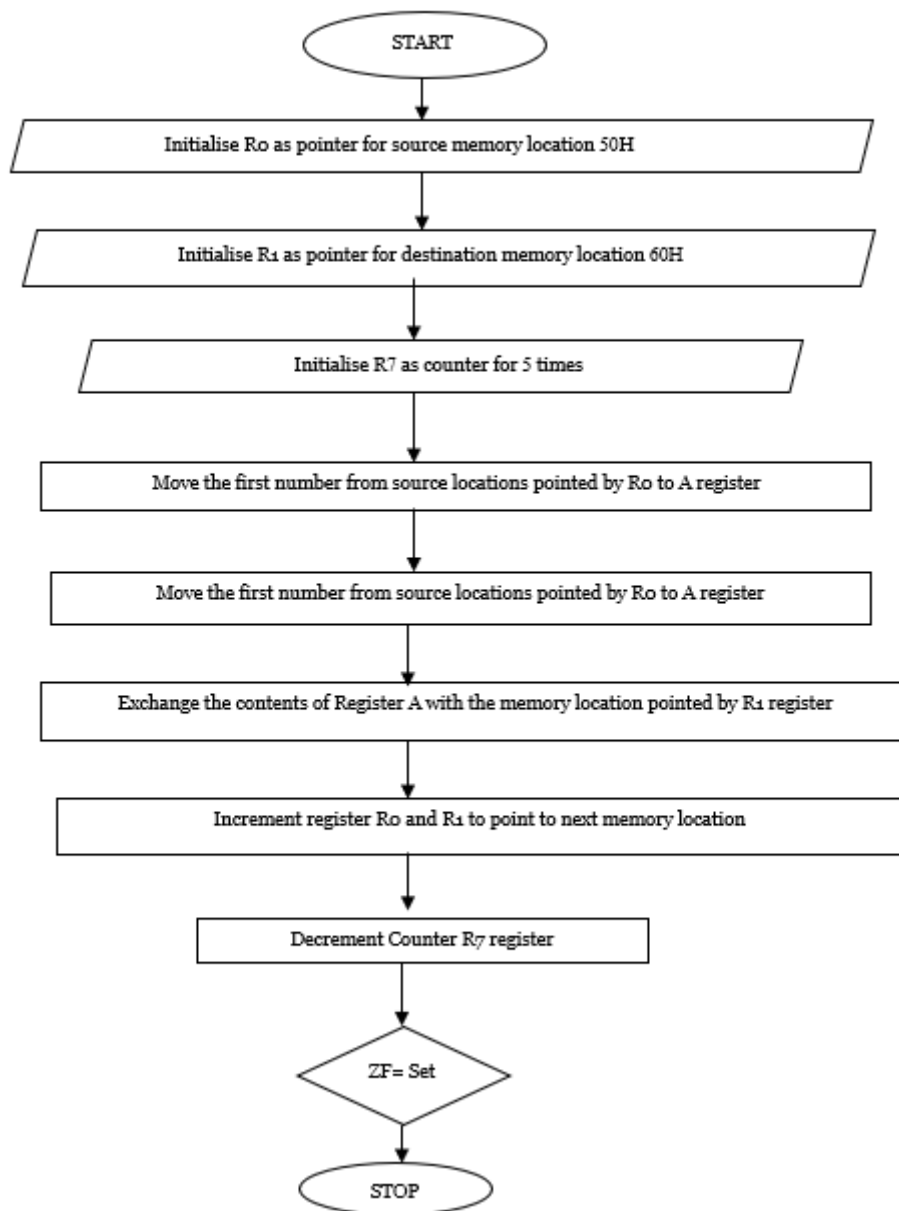


Fig 8.3 Flowchart for Block Exchange

Step 3- Assembly Language Sample Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 00H	
C:0x0000	7850		MOV R0,#50H	Initialize memory pointer R0 as a source
C:0x0002	7960		MOV R1, #60H	Initialize memory pointer R1 as a destination
C:0x0004	7F05		MOV R7, #05H	Initialize counter.
C:0x0006	E6	UP	MOV A, @R0	Move the content of first location in to accumulator.
C:0x0007	C7		XCH A, @R1	Exchange with destination memory
C:0x0008	F6		MOV @R0,A	Move the contents of Accumulator to source locations
C:0x0009	08		INC R0	Increment source location
C:0x0009	09		INC R1	Increment destination location
C:0x000A	DFF9		DJNZ R7, UP	Decrement iteration count and if not zero jump to step 5.
			END	

Input and Output Window:

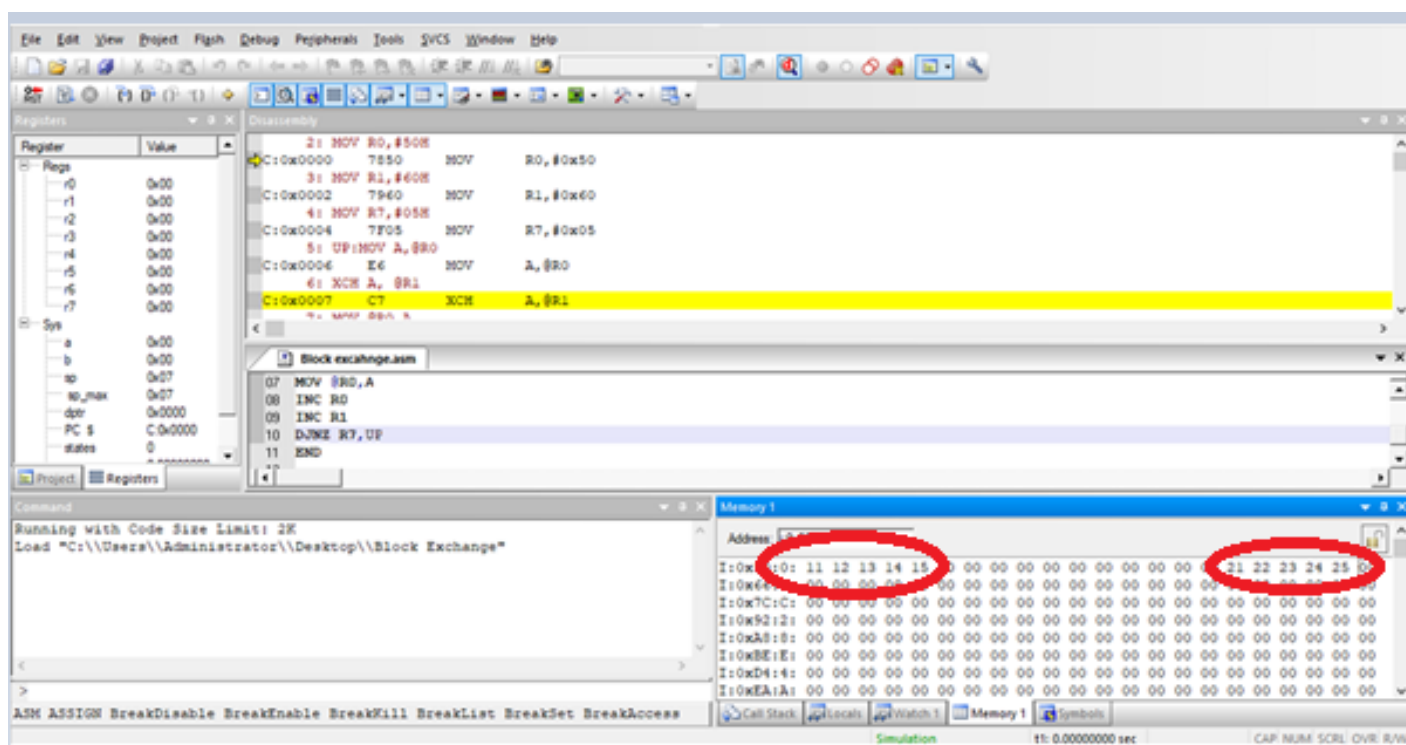


Fig 8.4 Before Execution Window

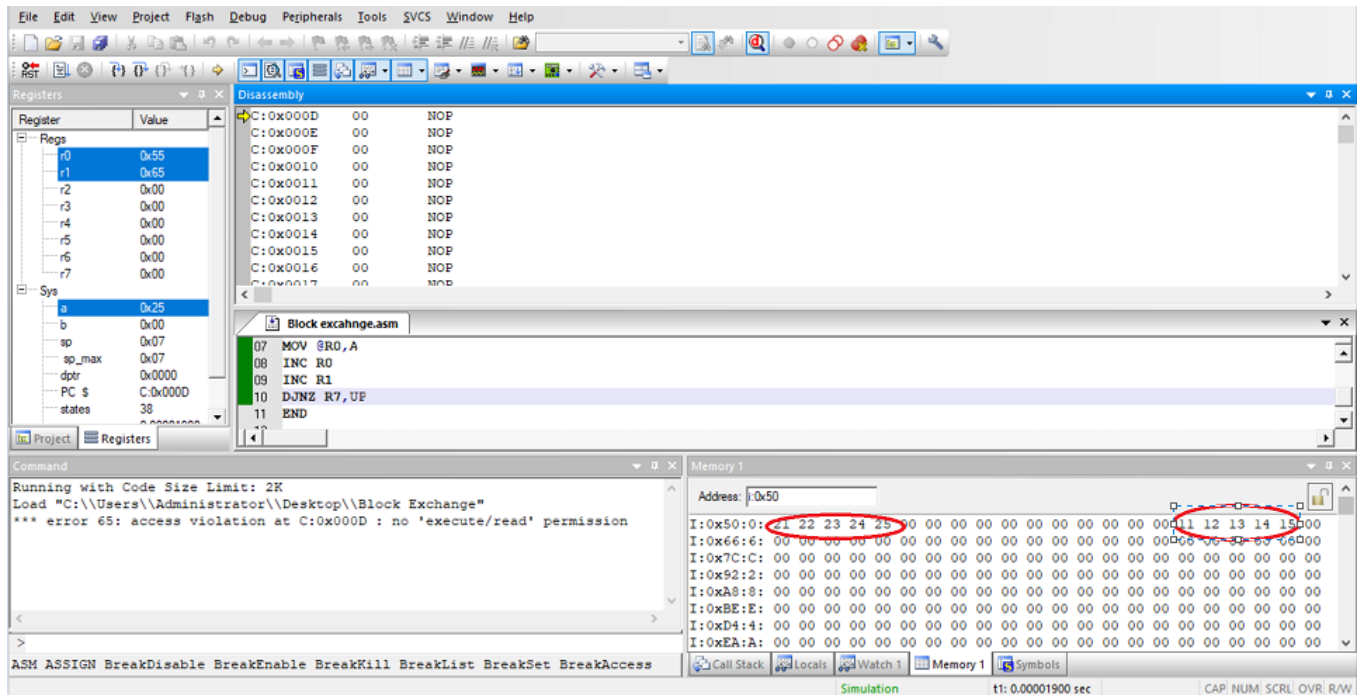


Fig 8.5 After Execution Window

Problem statement 1 for student: Write a program to perform Block Exchange from source to destination using external memory locations.

Step 1-Algorithm	Step 2-Flowchart

XI Observations for sample program (use blank sheet provided if space not sufficient)

Before execution				After execution			
Memory Address	Contents	Memory Address	Contents	Memory Address	Contents	Memory Address	Contents

XV Results (Output of the Program)

.....

.....

.....

.....

XVI Interpretation of Results (Give meaning of the above obtained results)

.....

.....

.....

.....

XVII Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....

.....

.....

.....

XIX References / Suggestions for further reading

1. <https://www.refreshnotes.com/2016/04/8051-program-exchange-block-of-data.html>
2. <https://www.tutorialspoint.com/program-branch-group-in-8051>
3. <https://www.codesexplorer.com/2016/12/8051-alp-to-move-data-from-internal-to-external.html>

XX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40% (10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 % (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 9: Finding the smallest number from the given data bytes

I Practical Significance

Microcontrollers use compare operation to test or compare register values. The results of these operations can update flag bits, which can then be used to change program flow through conditional execution. This practical will help the students to develop skills to use the compare and loop instructions to find smallest number from a block of data bytes.

II Industry/Employer Expected Outcome(s)

Maintain microcontroller based systems.

III Course Level Learning Outcome(s)

Develop program in 8051 in assembly language for the given operation.

IV Laboratory Learning Outcome(s)

Develop an ALP for identifying smallest number from the given data bytes stored in internal/external data memory

V Relevant Affective Domain related outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

In 8051 microcontroller, instructions CJNE and DJNZ can be used in combination for finding smallest number from a block of data bytes.

CJNE : Compare and Jump If Not Equal -

CJNE compares the value of two operands and branches to the indicated relative address if operands are not equal. If the two operands are equal program flow continues with the instruction following the CJNE instruction. CY flag is affected. CY flag is set if the first operand is smaller than second operand else it is reset.

Mnemonic	Description
CJNE A,direct,rel	Compares direct byte to the accumulator and jumps if not equal.
CJNE A,#data,rel	Compares immediate data to the accumulator and jumps if not equal
CJNE Rn,#data,rel	Compares immediate data to the register and jumps if not equal.
CJNE @Ri,#data,rel	Compares immediate data to indirect register and jumps if not equal.

DJNZ Rn, relative

This instruction decrements the contents of register by 1 and jump to the relative address if not zero.

VII Required Resources/apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1.	Desktop PC	Loaded with open-source IDE, simulation and program downloading software.	1 No.

VIII Precautions to be followed

- 1) Check rules / syntax of assembly language programming.

IX Procedure**Develop Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL or INTEL and select 80c51AH or AT89C51.
5. Type the program in text editor and save as .asm or .a51.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window. It will display all internal registers of 8051 and their contents.
11. Note down the readings in observation table

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write a program to find smallest number from the given array of 05 data bytes stored in internal RAM locations 30H onwards and store the smallest number in location 40H.

Step 1: Algorithm

1. Initialize the comparison count to 04H which is number of data bytes minus one.
2. Initialize source memory pointer R0 to 30H.
3. Move the contents of source location pointed by R0 to Accumulator
4. Increment source memory pointer
5. Move the contents of source location to register B
6. Compare the two numbers.
7. If number in A is less than number in B, then go to step 9.

8. Move smallest number from register B to register A
9. Decrement comparison count by 1.
10. If count is not zero go to step 4.
11. Store the smallest number from register A location 40H

Step 2- Flow Chart

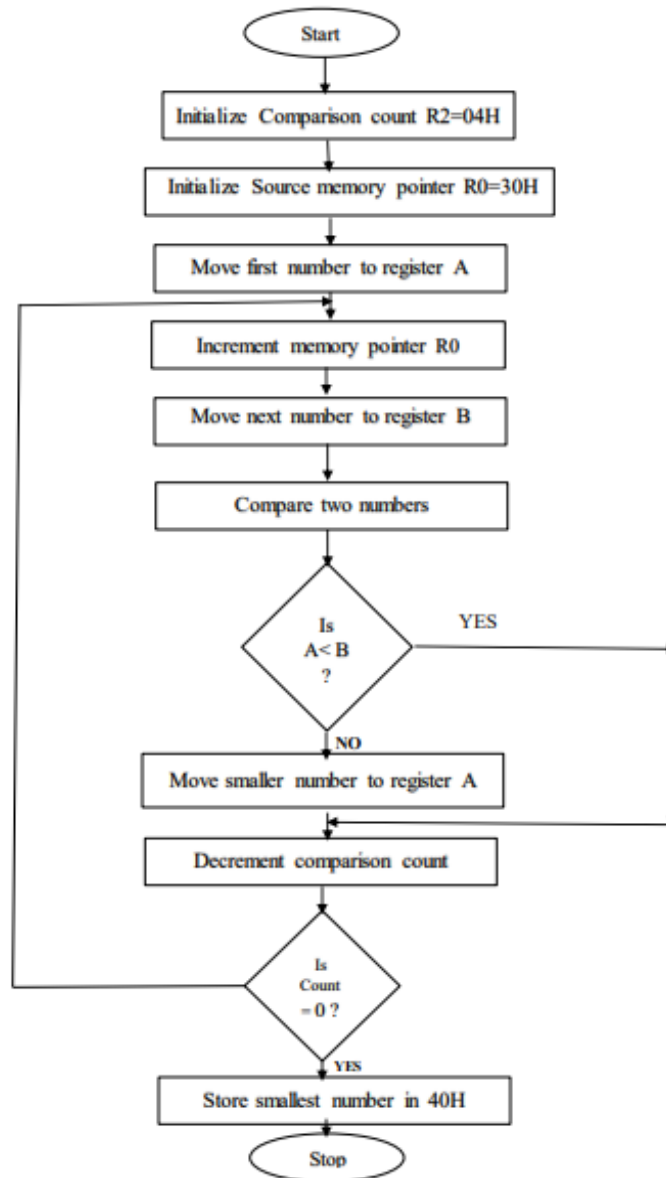


Fig 5.1 Flowchart to find smallest number

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0x0000	7A04		MOV R2,#04H	;Initialize Comparison count
C:0x0002	7830		MOV R0,#30H	;Initialize Source memory pointer R0=30H

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x0004	E6		MOV A,@R0	;Move first number to register A
C:0x0005	08	UP:	INC R0	;Increment memory pointer
C:0x0006	86F0		MOV B,@R0	;Move next number to register B
C:0x0008	B5F0000		CJNE A,B, DOWN	;Compare two numbers
C:0x000B	4002	DOWN:	JC SMALL	;If number in register A is Smaller than number in register B, jump DOWN
C:0x000D	E5F0		MOV A,B	;Move smaller number in register B to A
C:0x000F	DAF4	SMALL	DJNZ R2,UP	;Decrement comparison count, if count \neq 0, jump UP
C:0x0011	F540		MOV 40H, A	;Move smallest number to location 40h
			END	

Output Window

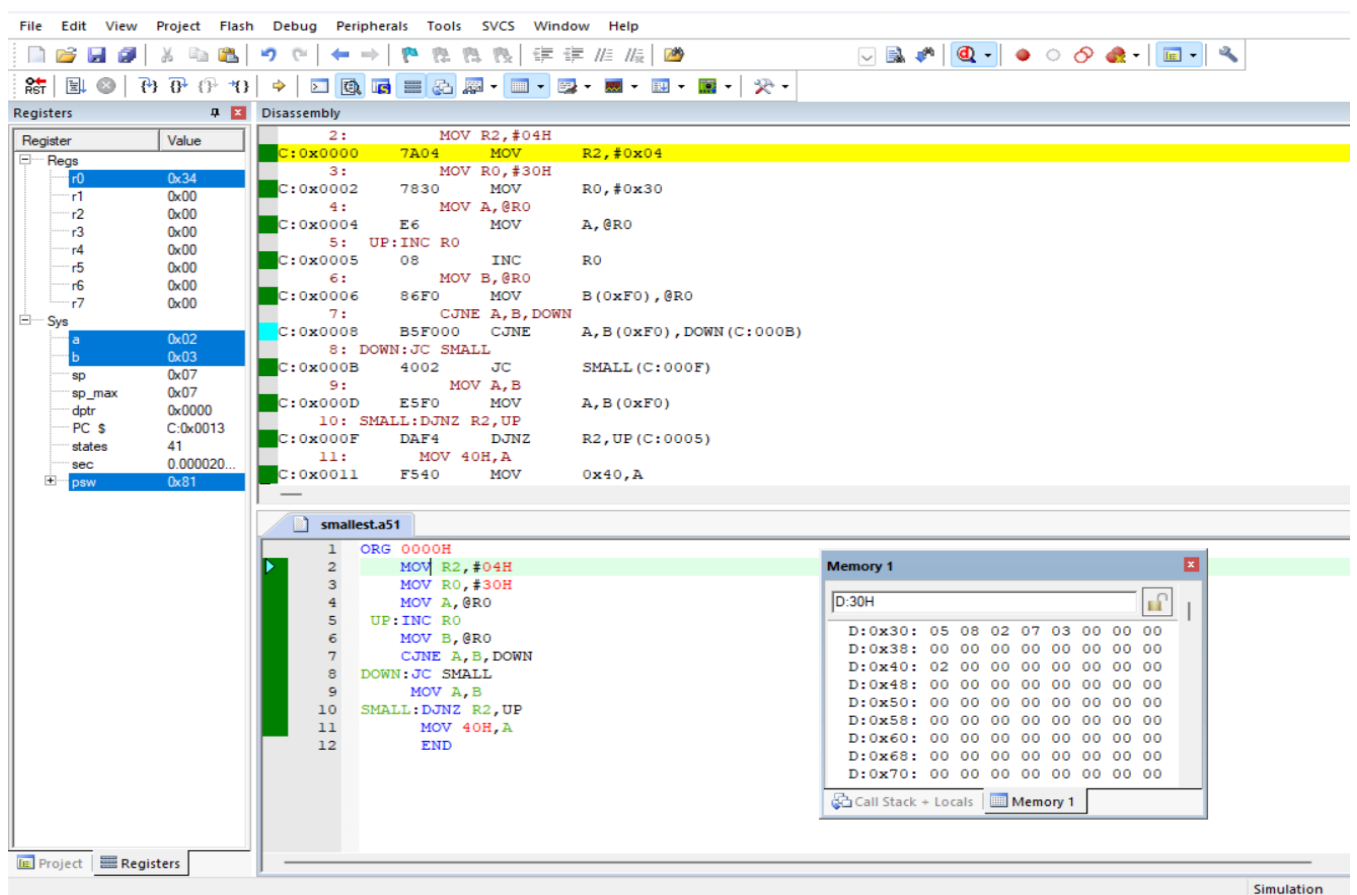


Fig 5.2 Output Window

Problem statement for student: Write a program to find smallest number from the given array of 05 data bytes stored in External RAM locations 3000H onwards and store the smallest number in location 4000H.

Step 1-Algorithm	Step 2-Flowchart

XII Observations for problem statement (use blank sheet provided if space not sufficient)

Before execution		After execution	
Memory location	Data	Memory location	Data
3000H		4000H	
3001H			
3002H			
3003H			
3004H			

XIII Results (Output of the Program)

.....

XIV Interpretation of Results (Give meaning of the above obtained results)

.....

XV Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....

XVI Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifies CO.

1. State the significance of counter in finding smallest number from series of numbers.
2. Explain the internal operation performed by CPU while executing CJNE instruction. Also state the effect on CY flag
3. Give the status of CY Flag when following instructions are executed:

```
MOV A, #05H
GO: CJNE A, #05H, GO
END
```


.....

.....

.....

.....

.....

.....

XVII References/Suggestions for further reading

1. https://josephscollege.ac.in/lms/Uploads/pdf/material/Instruction_set_of_Microcontroller_8051.pdf
2. <https://www.refreshnotes.com/2016/04/8051-program-smallest-element-in-array.html>
3. <https://www.daenotes.com/electronics/digital-electronics/8051-microcontroller-instruction-types>
4. https://www.keil.com/support/man/docs/a51/a51_cjne.htm

XVIII Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 10: Finding the Largest number from the given data bytes

I Practical Significance

Microcontrollers use compare operation to test or compare register values. The results of these operations can update flag bits, which can then be used to change program flow through conditional execution. This practical will help the students to develop skills to use the compare and loop instructions to find largest number from a block of data bytes.

II Industry/Employer Expected Outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Develop program in 8051 in assembly language for the given operation.

IV Laboratory Learning Outcome(s)

Develop an ALP for identifying largest number from the given data bytes stored in internal/external data memory

V Relevant Affective Domain related outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

In 8051 microcontroller, instructions CJNE and DJNZ can be used in combination for finding smallest number from a block of data bytes.

CJNE : Compare and Jump If Not Equal -

CJNE compares the value of two operands and branches to the indicated relative address if operands are not equal. If the two operands are equal program flow continues with the instruction following the CJNE instruction. CY flag is affected. CY flag is reset if the first operand is Larger than second operand. Carry flag status can be checked using instructions JC and JNC.

JC: Jump if Carry Set

Syntax: JC relative address

This instruction will branch to the address indicated by relative address if the Carry flag is set. If the Carry flag is not set program execution continues with the instruction following the JC instruction.

JNC: Jump if Carry is not Set

Syntax: JNC relative address

This instruction will branch to the address indicated by relative address if the Carry flag is not set.

If the Carry flag is set program execution continues with the instruction following the JC instruction.

VII Required Resources/apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1.	Desktop PC	Loaded with open-source IDE, simulation and program downloading software.	1 No.

VIII Precautions to be followed

- 1) Check rules / syntax of assembly language programming.

IX Procedure**Develop Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL or INTEL and select 80c51AH or AT89C51.
5. Type the program in text editor and save as .asm or .a51.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window. It will display all internal registers of 8051 and their contents.
11. Note down the readings in observation table

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write a program to find Largest number from the given array of 05 data bytes stored in external RAM locations 3000H onwards and store the Largest number in location 3010H.

Step 1: Algorithm

1. Initialize the comparison count to 04H which is number of data bytes minus one.
2. Initialize external source memory pointer DPTR to 3000H.
3. Move the contents of external source location pointed by DPTR to Accumulator
4. Move data in register A to register B
5. Increment source memory pointer
6. Move the contents of source location to register A
7. Compare the two numbers.
8. If number in A is larger than number in B, then go to step 10.

9. Move largest number from register B to register A
10. Decrement comparison count by 1.
11. If count is not zero go to step 4.
12. Store the smallest number from register A location 3010H

Step 2: Flowchart Step 1: Algorithm

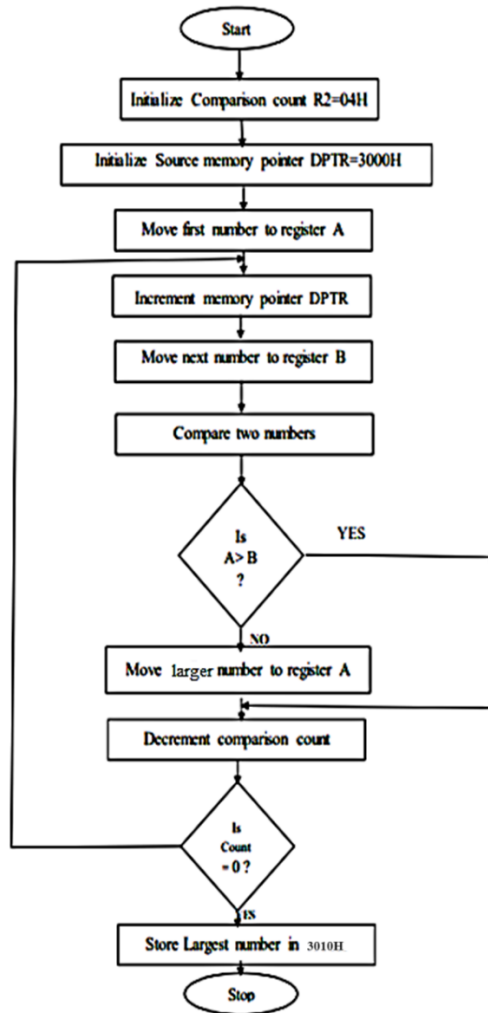


Fig 10.1 Flowchart to find Largest number

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0x0000	7A04		MOV R2,#04H	;Initialize Comparison count
C:0x0002	903000		MOV DPTR,#3000H	;Initialize Source memory pointer R0=30H
C:0x0005	E0		MOVX A,@DPTR	;Move first number to register B
C:0x0006	F5F0	UP:	MOV B,A	
C:0x0008	A3		INC DPTR	;Increment memory pointer

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x0009	E0		MOVX A,@DPTR	;Move next number to register A
C:0x000A	B5F000		CJNE A,B, DOWN	;Compare two numbers
C:0x000D	5002	DOWN:	JNC LARGE	;If number in register A is Smaller than number in register B, jump DOWN
C:0x000F	E5F0		MOV A,B	;Move larger number in register B to A
C:0x0011	DAF3	LARGE:	DJNZ R2,UP	;Decrement comparison count, if count \neq 0, jump UP
C:0x0013	903010		MOV DPTR,#3010H	
C:0x0016	F0		MOVX @DPTR, A	;Move smallest number to location 3010H
C:0x00017	80FE	HERE:	SJMP HERE	;Stop
			END	

Output Window

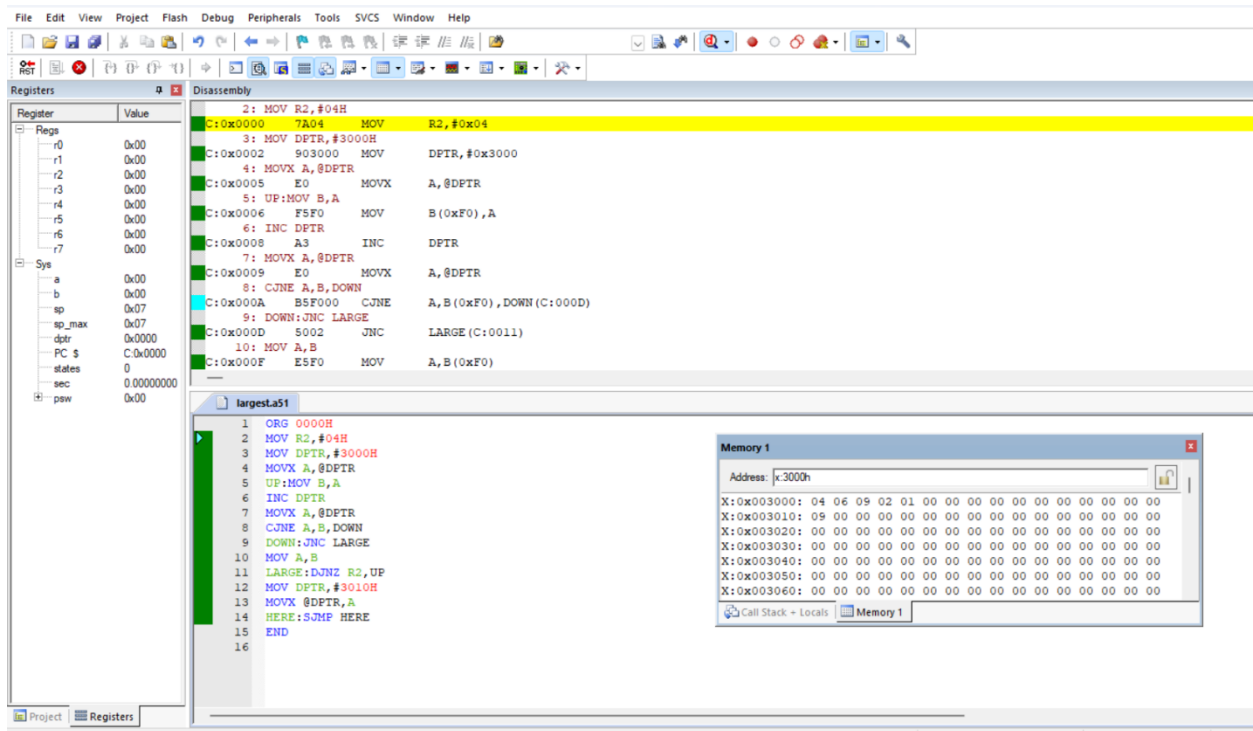


Fig 10.2 Output Window

Problem statement for student: Write a program to find Largest number from the given array of 10 data bytes stored in internal RAM location 50H onwards and store the Largest number in location 60H.

Step 1-Algorithm	Step 2-Flowchart

Step 3- Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments

X Resources used

Sr. No.	Name of Resource	Specifications	Quantity

XI Actual Procedure Followed (use blank sheet provided if space not sufficient)

1.
2.
3.
4.
5.
6.
7.
8.

XII Observations for problem statement (use blank sheet provided if space not sufficient)

Before execution		After execution	
Memory location	Data	Memory location	Data
50H		60H	
51H			
52H			
53H			
54H			
55H			
56H			
57H			
58H			
59H			

XIII Results (Output of the Program)

.....

XIV Interpretation of Results (Give meaning of the above obtained results)

.....

XV Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....

XVI Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifies CO.

1. State the function of DPTR register.
2. Explain operation of instruction MOVX @DPTR, A

[Space for Answers]

.....

XVII References/Suggestions for further reading

1. <https://electronicsforyou.in/8051-program-to-find-the-largest-number-in-an-array/>
2. https://josephscollege.ac.in/lms/Uploads/pdf/material/Instruction_set_of_Microcontroller_8051.pdf
3. <https://www.refreshnotes.com/2016/04/8051-program-smallest-element-in-array.html>
4. <https://www.daenotes.com/electronics/digital-electronics/8051-microcontroller-instruction-types>

XVIII Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 11: Arranging numbers in Ascending order

I Practical Significance

Sorting is any process of arranging information systematically in ascending or descending order. This allows us to write better programs like indexing to fetch the information faster, allows faster search techniques, removes duplicate information and has many uses in statistical applications. Arranging numbers in ascending order is critical for highlighting significant values, optimizing processes, and enhancing the effectiveness of data analysis and presentation. This practical will help the students to develop skills to understand how to access data from external memory, use of branch instructions and arranging numbers in ascending order.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Develop program in 8051 in assembly language for the given operation.

IV Laboratory Learning Outcome(s)

Develop an ALP for arranging numbers in ascending order stored in internal/external data memory.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

Ascending order/Descending order

The block of data consists of numbers in random order, to arrange these numbers in ascending or descending order bubble sort method is used.

If the given block of data has to be sorted in ascending order, then bubble sort will start by comparing the first element of the block with the second element, if the first element is greater than the second element, it will swap both the elements, and then move on to compare the second and the third element, and so on.

The different types of counters used in this process:

1. Byte counter –to access data from block of data
2. Pass counter –to repeat this comparison are required to arrange the numbers in ascending or descending order.

Branch instructions like

JNC—Jump if no carry

CJNE—Compare and jump if not equal to are used to sort the numbers.

VII Resources Required

Sr. No.	Instrument /Components	Specification	Quantity
1	Desktop PC	Loaded with open-source IDE, simulation and program downloading software	1 No.

VIII Precautions to be Followed

1. Check rules / syntax of assembly programming.

IX Procedure**Write Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write and execute a program to arrange the ten data values in external memory in descending order**Step 1-Algorithm**

Step 1: Initialize a counter for comparison (Pass counter).

Step2: Initialize memory pointer to read number from the array.

Step3: Initialize byte counter.

Step 4: Read numbers from array.

Step 5: Compare two numbers.

- Step 6: If number less than or equal to next number, then go to step 8.
 Step 7: Replace number with next number which is largest.
 Step 8: Increment memory pointer to read next number in the array.
 Step 9: Decrement byte counter by 1.
 Step 10: If byte counter is not equal to zero then go to step 4.
 Step 11: Decrement pass counter by 1.
 Step 12: If pass counter is not equal to zero then go to step 2.
 Step 13: Stop.

Step 2-Flow Chart:

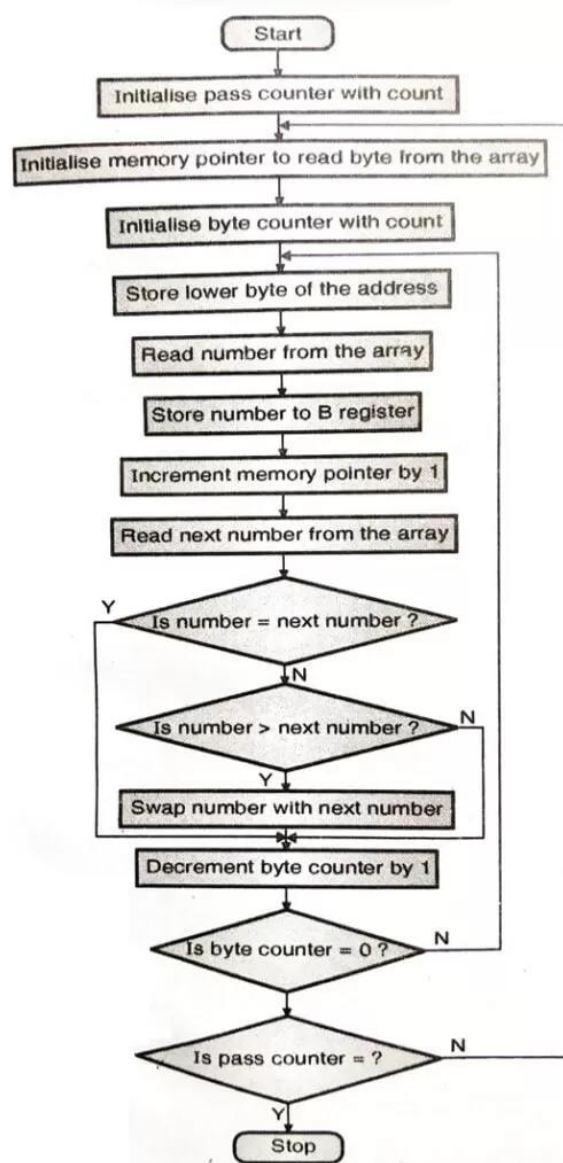


Fig 11.1: Flowchart for Descending order

Step 3- Assembly Language Sample Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000h	
C:0x0000	7805		MOV R0, #0AH	;Initialize pass counter
C:0x0002	904000	REP1:	MOV DPTR, #4000H	;Initialize memory pointer
C:0x0005	7904		MOV R1, #09H	;Initialize byte counter
C:0x0007	AA82	REPEAT	MOV R2, DPL	;Save the lower byte address
C:0x0009	E0		MOVX A, @DPTR	;Read number from array
C:0x000A	F5F0		MOV 0F0H, A	;Transfer the number to B register
C:0x000C	A3		INC DPTR	;Increment memory pointer
C:0x000D	E0		MOVX A, @DPTR	;Read next number from array
C:0x000E	B5F002		CJNE A, 0F0H, NEXT	;Compare number with next number
C:0x0011	011C		AJMP SKIP	
C:0x0013	5007	NEXT:	JNC SKIP	;If number>next number then go to SKIP
C:0x0015	8A82		MOV DPL, R2	;Else exchange the number with next number
C:0x0017	F0		MOVX @DPTR, A	
C:0x0018	A3		INC DPTR	
C:0x0019	E5F0		MOV A, 0F0H	
C:0x001B	F0		MOVX @DPTR, A	
C:0x001C	D9E7	SKIP:	DJNZ R1, REPEAT	;Decrement byte and if count byte is not zero go to Up
C:0x001E	D8E0		DJNZ R0, REP1	;Decrement pass counter and if not zero go to UP1
			END	

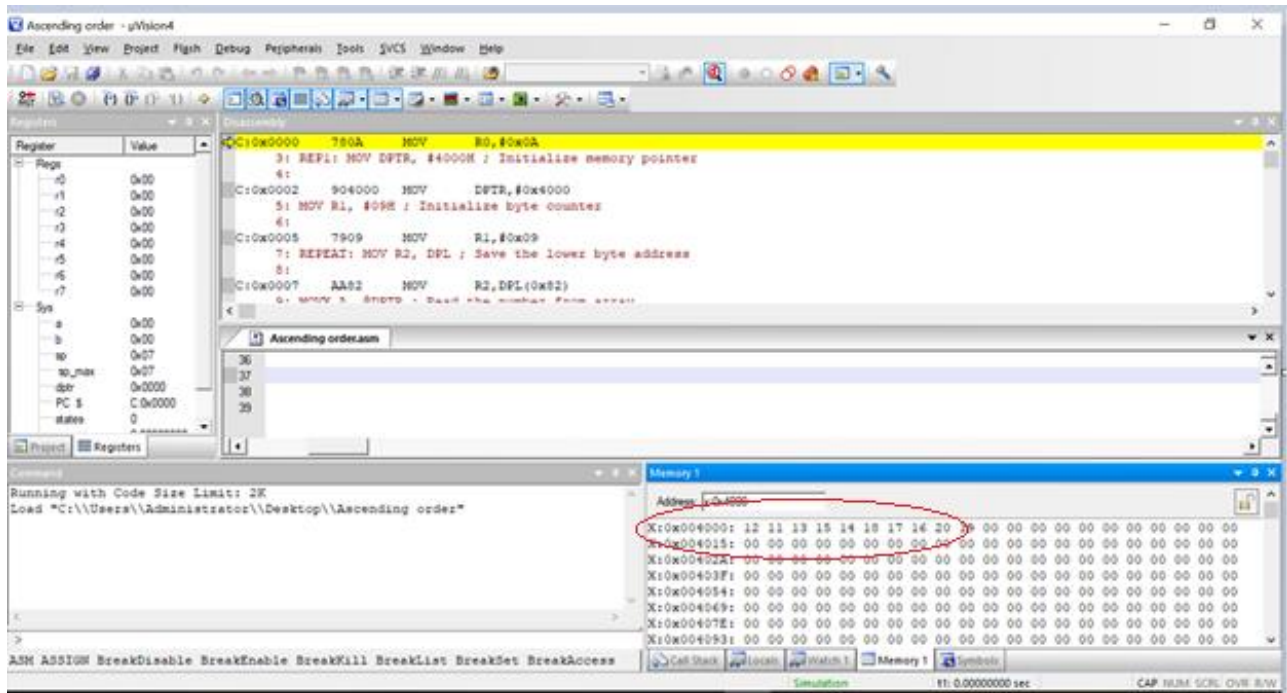


Fig 11.2 Input Window

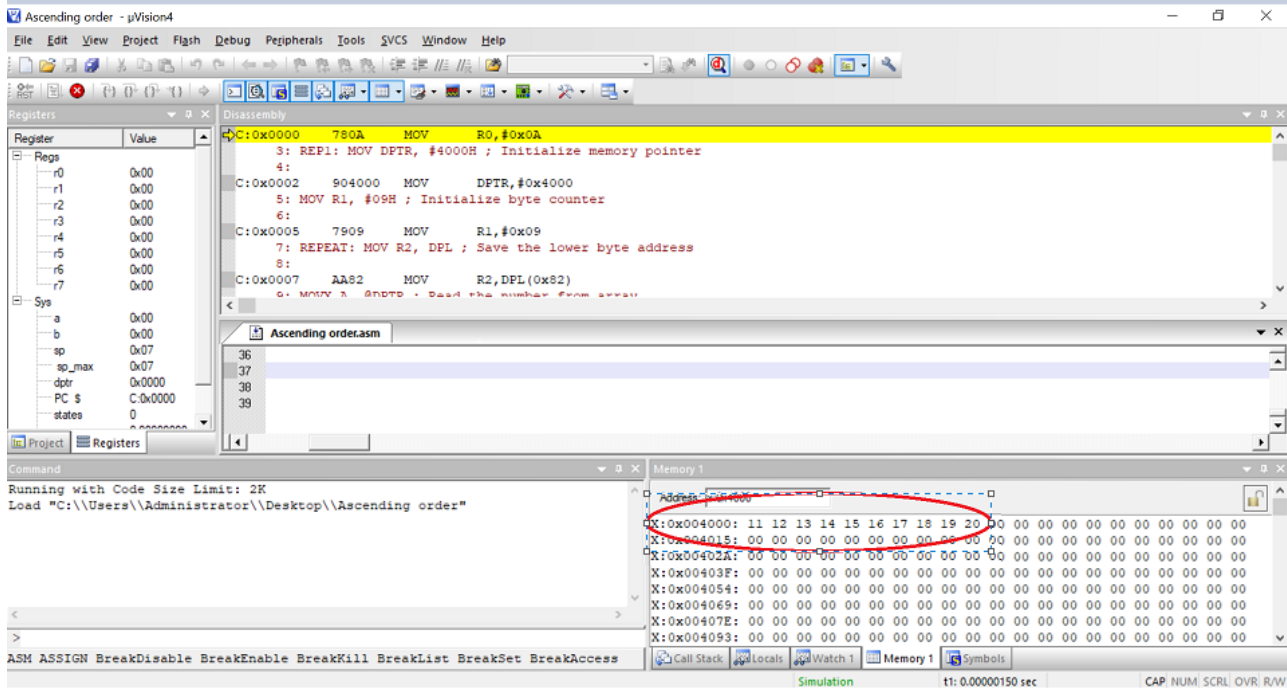


Fig 11.3 Output Window

Problem statement 1 for student: Write a program to arrange the numbers in ascending order. Assume internal memory locations.

<p>Step 1-Algorithm</p> 	<p>Step 2-Flowchart</p>
--	--

Step 3- Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments

Memory Address	Hex Code	Label	Mnemonics	Comments

X Resources Used:

S. No.	Instrument /Components	Specification	Quantity

XI Actual Procedure Followed (use blank sheet provided if space not sufficient)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XI Observations for sample program (use blank sheet provided if space not sufficient)

Before execution		After execution	
Memory location	Data	Memory location	Data

XV Results (Output of the Program)

.....

.....

.....

.....

XVI Interpretation of Results (Give meaning of the above obtained results)

.....

.....

.....

.....

XVII Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....

.....

.....

.....

.....

.....

XIX References / Suggestions for further reading

1. <https://electronicsforu.in/8051-program-to-arrange-numbers-in-ascending-order/>
2. <https://technobyte.org/branching-instructions-8051/>
3. <https://instrumentationforindustry.com/programming-8051-microcontroller-sorting-arrays-algorithm/>

XX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40% (10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 % (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 12: Arranging numbers in Descending order

I Practical Significance

Sorting is any process of arranging information systematically in ascending or descending order. Helps in decision-making processes where resources are to be allocated based on priority, with the most critical or valuable cases handled first. Arranging numbers in descending order is critical for highlighting significant values, optimizing processes, and enhancing the effectiveness of data analysis and presentation. This practical will help the students to develop skills to understand how to access data from external memory and use of branch instructions.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Develop program in 8051 in assembly language for the given operation.

IV Laboratory Learning Outcome(s)

Develop an ALP for arranging numbers in descending order stored in internal/external data memory.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

Ascending order/Descending order

The block of data consists of numbers in random order, to arrange these numbers in ascending or descending order bubble sort method is used.

If the given block of data has to be sorted in descending order, then bubble sort will start by comparing the first element of the block with the second element, if the first element is smaller than the second element, it will swap both the elements, and then move on to compare the second and the third element, and so on.

The different types of counters used in this process:

1. Byte counter –to access data from block of data
2. Pass counter –to repeat this comparison are required to arrange the numbers in ascending or descending order.

Branch instructions like

JNC—Jump if no carry

CJNE—Compare and jump if not equal to are used to sort the numbers.

VII Resources Required

Sr. No.	Instrument /Components	Specification	Quantity
1	Desktop PC	Loaded with open-source IDE, simulation and program downloading software	1 No.

VIII Precautions to be Followed

1. Check rules / syntax of assembly programming.

IX Procedure**Write Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write and execute a program to arrange the ten data values in external memory in descending order**Step 1-Algorithm****Algorithm:**

Step 1: Initialize a counter for comparison (Pass counter).

Step2: Initialize memory pointer to read number from the array.

- Step 3:** Initialize byte counter.
- Step 4:** Read numbers from array.
- Step 5:** Compare two numbers.
- Step 6:** If number greater than or equal to next number, then go to step 8.
- Step 7:** Replace number with next number which is smaller.
- Step 8:** Increment memory pointer to read next number in the array.
- Step 9:** Decrement byte counter by 1.
- Step 10:** If byte counter is not equal to zero then go to step 4.
- Step 11:** Decrement pass counter by 1.
- Step 12:** If pass counter is not equal to zero then go to step 2.
- Step 13:** Stop.

Step 2-Flow Chart:

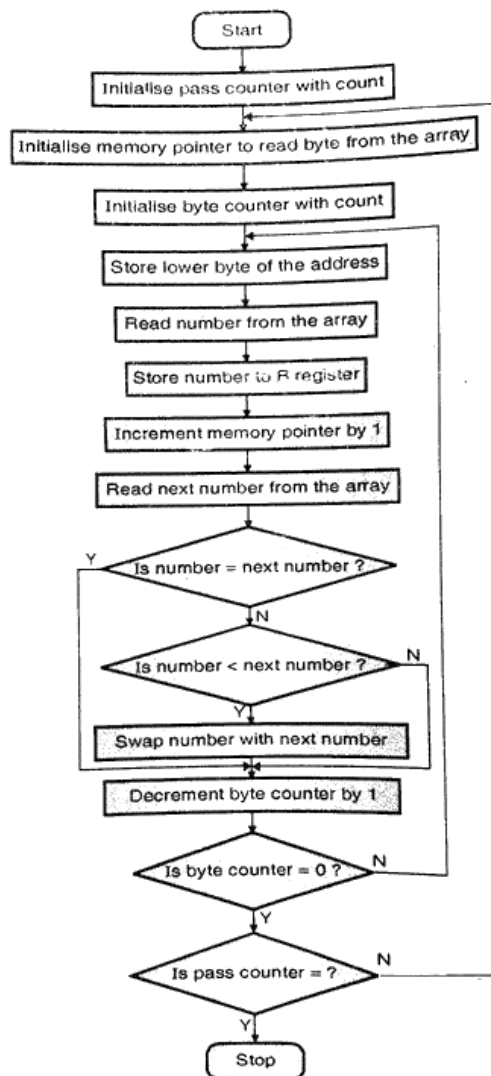


Fig 12.1: Flowchart for Descending order

Step 3- Assembly Language Sample Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 00H	
C:0X0000	780A		MOV R0, #0AH	Initialize pass counter.
C:0X0002	904000	REP1	MOV DPTR, #4000H	Initialize memory pointer
C:0X0005	7909		MOV R1, #09H	Initialize byte counter
C:0X0007	AA82	REPEAT	MOV R2, DPL	Save the lower byte address
C:0X0009	E0		MOVX A, @DPTR	Read the number from array
C:0X000A	F5F0		MOV 0F0H, A	Store the number in register B
C:0X000C	A3		INC DPTR	Increment memory pointer
C:0X000D	E0		MOVX A, @DPTR	Take the next number from array
C:0X000E	B5F002		CJNE A, 0F0H, NEXT	Compare number with next number
C:0X0011			AJMP SKIP	Jump to SKIP unconditionally
C:0X0013	4007	NEXT:	JC SKIP	If number < next number then go to skip
C:0X0015	8A82		MOV DPL, R2	Else exchange the number with next number
C:0X0017	F0		MOVX@DPTR, A	Copy greater number to memory locations
C:0X0018	A3		INC DPTR	Increment memory pointer
C:0X0019	E5F0		MOV A, 0F0H	
C:0X001B	F0		MOVX @DPTR, A	
C:0X001C	D9E9	SKIP:	DJNZ R1, REPEAT	Decrement byte counter by 1, if byte counter ≠ 0 then go to REPEAT.
C:0X001E	D8E2		DJNZ R0, REP1	Decrement pass counter if not zero then go to REP1
C:0X0020	0120	STOP:	AJMP STOP	STOP
			END	

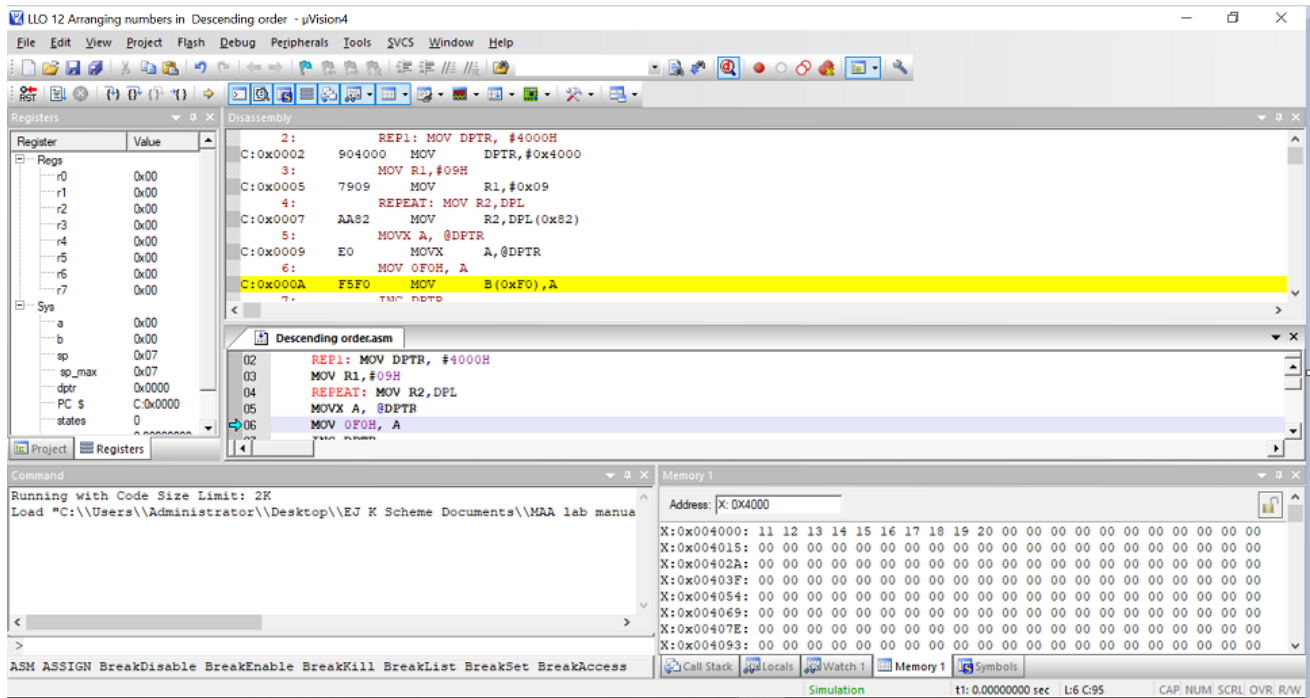


Fig 12.2 Input Window

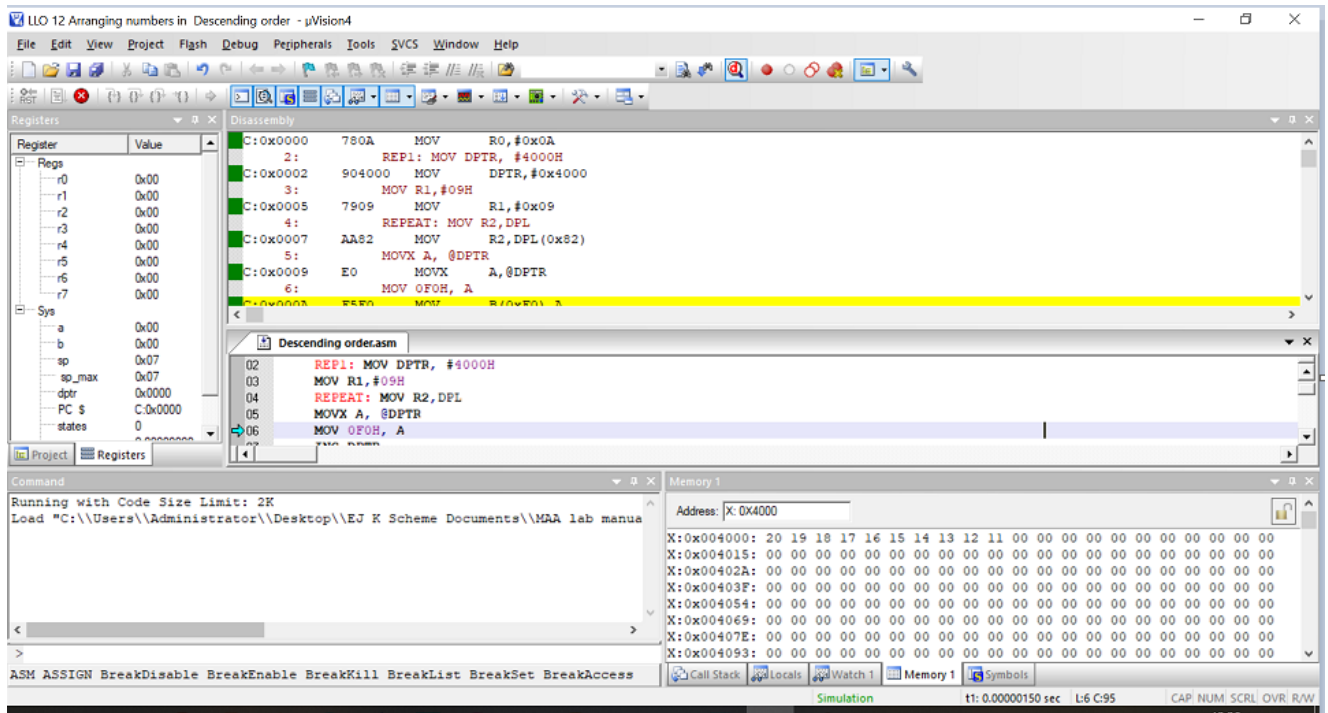


Fig 12.3 Output Window

.....

.....

.....

.....

XIX References / Suggestions for further reading

1. <https://electronicsforyou.in/8051-program-to-arrange-numbers-in-descending-order/>
2. <https://technobyte.org/branching-instructions-8051/>
3. <https://instrumentationforindustry.com/programming-8051-microcontroller-sorting-arrays-algorithm/>

XX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40% (10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 % (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 13: Generate delay using timer register

I Practical Significance

Generation of time delay is most important concept in embedded systems. The purpose of the delay program is to introduce a pause or delay in program execution for a specified amount of time. Most of the times, precise time delay needs to be generated between two actions in microcontroller applications, like blinking of LED, Pulse generation, monitoring switch etc. We can generate the time delay using the Loop technique. Precise time delay can be generated using inbuilt timers in microcontroller. This practical will help the students to develop skills to create the time delay by using timer registers and microcontroller crystal frequency.

II Industry/Employer Expected Outcome(s)

Maintain microcontroller based systems.

III Course Level Learning Outcome(s)

Develop program using timers and interrupts

IV Laboratory Learning Outcome(s)

Write an ALP to generate delay using timer register

V Relevant Affective Domain related outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

The 8051 microcontroller has two 16-bit timers/counters called Timer 0 and Timer1.

They can be used either as timers to generate a time delay or as counters to count the events.

Since the microcontroller 8051 has an 8-bit architecture, each 16-bit timer is accessed as two separate registers THx and TLx

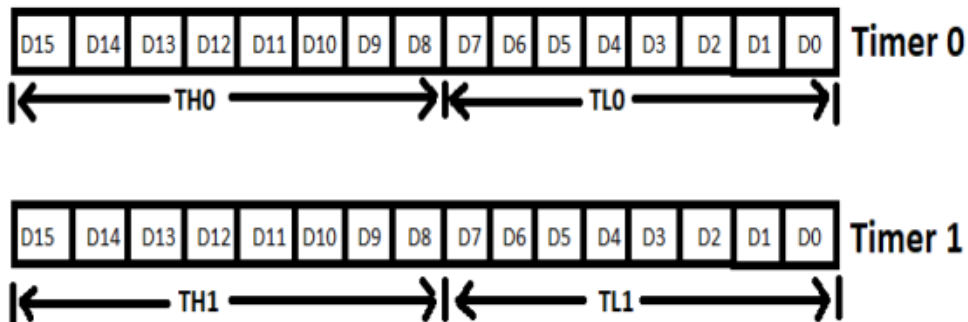


Fig 13.1 Timer 0 / Timer 1

TMOD Register:

TMOD register is used to select various timer operation modes for Timer 0 and Timer 1

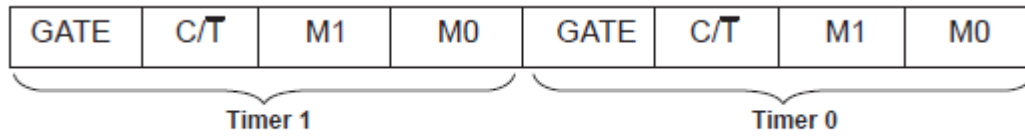


Fig 13.2 TMOD Register

GATE : When TRX (in TCON) is set and GATE=1 , TIMER/COUNTER will run only while INTX pin is high(hardware control), when GATE=0 , TIMER/COUNTER will run only while TRX=1 regardless of state of INTX pin (software control)

C/T: Timer or Counter Selector

1 = counter – external timing signal , input from T0/T1 pin

0 = timer – internal timing signal , input from internal system clock

M1 M0 : These two bits selects the Time / Counter operating mode.

M1	M0	MODE	DESCRIPTION
0	0	0	13-bit timer/counter
0	1	1	16 bit timer/counter
1	0	2	8 bit auto-reload
1	1	3	Split mode: (Timer 0) TL0 is an 8-bit timer/counter controlled by the standard timer 0 Control bits. TH0 is an 8-bit timer and is controlled by timer 1 control bits. Timer1 / counter1 is stopped

TCON Register:

TCON is 8 bit SFR used to control the Timer/Counter operations

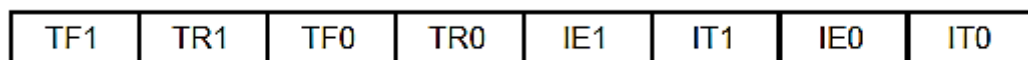


Fig 13.3 TCON Register

Bit	Symbol	TCON Bit Function
7	TF1	Timer 1 Overflow flag. Set when timer rolls from all 1's to 0. Cleared when processor vectors to execute interrupt service routine located at program address 001Bh.
6	TR1	Timer 1 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to halt timer.
5	TF0	Timer 0 Overflow flag. Set when timer rolls from all 1's to 0. Cleared when processor vectors to execute interrupt service routine located at program address 000Bh.

Bit	Symbol	TCON Bit Function
4	TR01	Timer 0 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to halt timer.
3	IE1 1	External interrupt 1 Edge flag. Set to 1 when a high-to-low edge signal is received on port 3.3 (INT1'). Cleared when processor vectors to interrupt service routine at program address 0013h. Not related to timer operations.
2	IT1 1	External interrupt 1 signal type control bit. Set to 1 by program to enable external interrupt 1 to be triggered by a falling edge signal. Set to 0 by program to enable a low-level signal on external interrupt 1 to generate an interrupt.
1	IE0 1	External interrupt 0 Edge flag. Set to 1 when a high-to-low edge signal is received on port 3.2 (INT0'). Cleared when processor vectors to interrupt service routine at program address 0003h. Not related to timer operations.
0	IT0 1	External interrupt 0 signal type control bit. Set to 1 by program to enable external interrupt 1 to be triggered by a falling edge signal. Set to 0 by program to enable a low-level signal on external interrupt 0 to generate an interrupt.

TIMER/COUNTER CONTROL LOGIC:

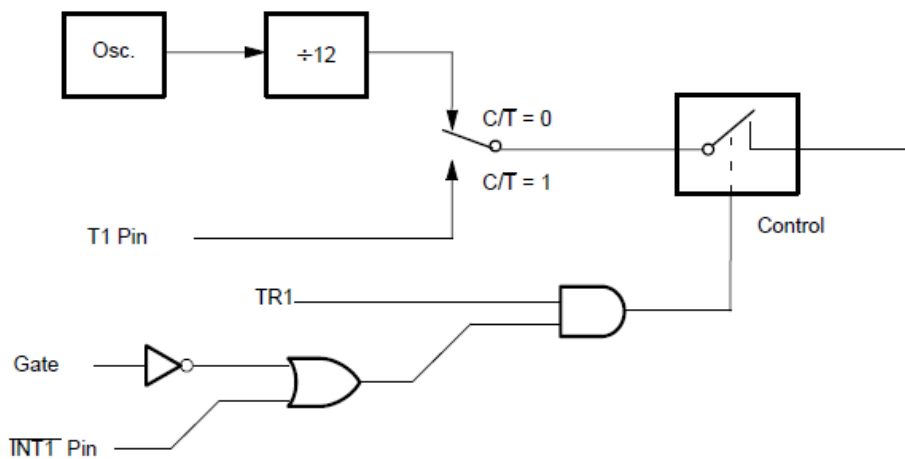


Fig 13.4 Timer/Counter Logic

TIME DELAY GENERATION USING TIMER REGISTERS:

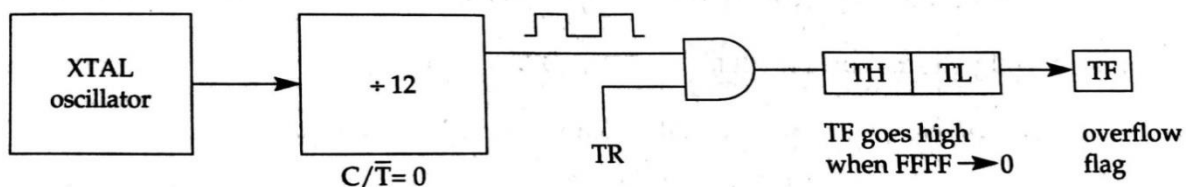


Fig 13.5 Timer mode 1 diagram

The Timer count increments by 1 at rate $F_{osc} / 12$

So time for one increment is $12 / F_{osc}$.

Therefore, if crystal frequency = 11.0592 MHz

$$\therefore \text{Counter frequency} = \frac{1}{12} \times 11.0592 \text{ MHz} = 921.6 \text{ kHz}$$

$$\therefore \text{Time for one increment} = 12 / \text{Fosc.} = (12 / 11.0592 \text{ MHz}) = 1.085 \mu\text{s}$$

$$\text{Time delay} = \text{Time for one increment} \times \text{number of increments till overflow (N)}$$

$$\text{Time delay} = (12 / \text{Fosc}) \times \text{number of increments (N)}$$

$$\text{Number of increments N till overflow} = (2^n - \text{Initial count})$$

Timer Mode	Value of n
Mode 0	13
Mode 1	16
Mode 2	8
Mode 3	8

10 ms Delay generation:

Required time delay = 10 ms

Oscillator frequency is Fosc. = 11.0592MHz

Time delay = (12 / Fosc) x number of increments (N)

$$10 \text{ ms} = (12 / 11.0592 \text{ MHz}) \times \text{number of increments (N)}$$

$$10 \text{ ms} = 1.085 \mu\text{sec.} \times N$$

$$N = (10 \times 10^3) \mu\text{sec.} / (1.085) \mu\text{sec.}$$

$$N = 9217$$

INITIAL COUNT CALCULATION:

$$\text{Initial count} = 2^n - N$$

Using TIMER MODE1 (n= 16)

$$\text{Initial count} = 2^{16} - N$$

$$= 65536 - 9217 = 56319 \text{ decimal} = \text{DBFF H}$$

THx = DBH ,

TLx = FFH

VII Required Resources/apparatus/equipment with specifications

Sr . No.	Instrument /Components	Specification	Quantity
1.	Desktop PC	Loaded with open source IDE, simulation and program downloading software.	1 No.

VIII Precautions to be followed

- 1) Check rules / syntax of assembly language programming.

IX Procedure

1. Write algorithm for given problem.
2. Draw flowchart for the same.
3. Develop assembly program using Integrated Development Environment (Keil IDE) or any other relevant software tool.
4. Debug program on IDE.
5. Execute program on IDE.
6. Create hex file for the program.

7. Download hex code in EPROM/Flash memory of microcontroller.
8. Connect CRO probe to port pin and observe waveform.
9. Measure ON time and OFF time on CRO and draw the same in observation Table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM : To toggle Port Pin P1.5 with 10ms delay using Timer

Step 1: Algorithm

1. Load the TMOD value register indicating which timer (Timer 0 or Timer 1) is to be used and which timer mode(0 or 1) is selected.
2. Load registers TL and TH with initial count values.
3. Start the timer.
4. Keep monitoring the timer flag (TF) with the “JNB TFx, target” instruction to see if it is raised. Get out of the loop when TF becomes high.
5. Stop the timer.
6. Toggle Port pin high and low
7. Clear the TF flag for the next round.
8. Go back to Step 2 to load TH and TL again.

Step 2: Flowchart

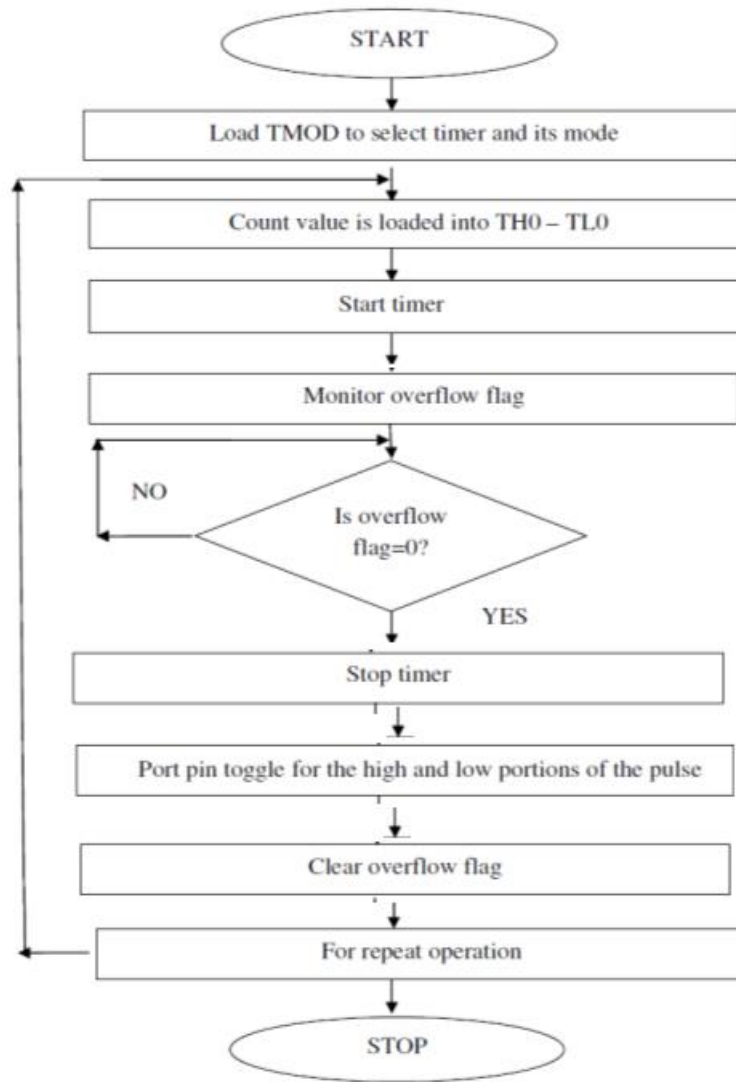


Fig 13.8 Flowchart for toggling port pin

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0x0000	758901		MOV TMOD ,#01H	;Timer 0, mode 1
C:0x0003	758AFF	REPEAT:	MOV TL0,#0FFH	;TL0=FFH
C:0x0006	758CDB		MOV TH0,#0DBH	;TH0=DBH
C:0x0009	D28C		SETB TR0	;start Timer 0
C:0x000B	308DFD	AGAIN:	JNB TF0, AGAIN	;monitor Timer 0 overflow flag until it rolls over
C:0x000E	C28C		CLR TR0	;stop Timer 0
C:0x0010	B295		CPL P1.5	;toggle P1.5

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x0012	C28D		CLR TFO	;clear Timer 0 flag
C:0x0014	80ED		SJMP REPEAT	;Reload TH, TL again

Output Window

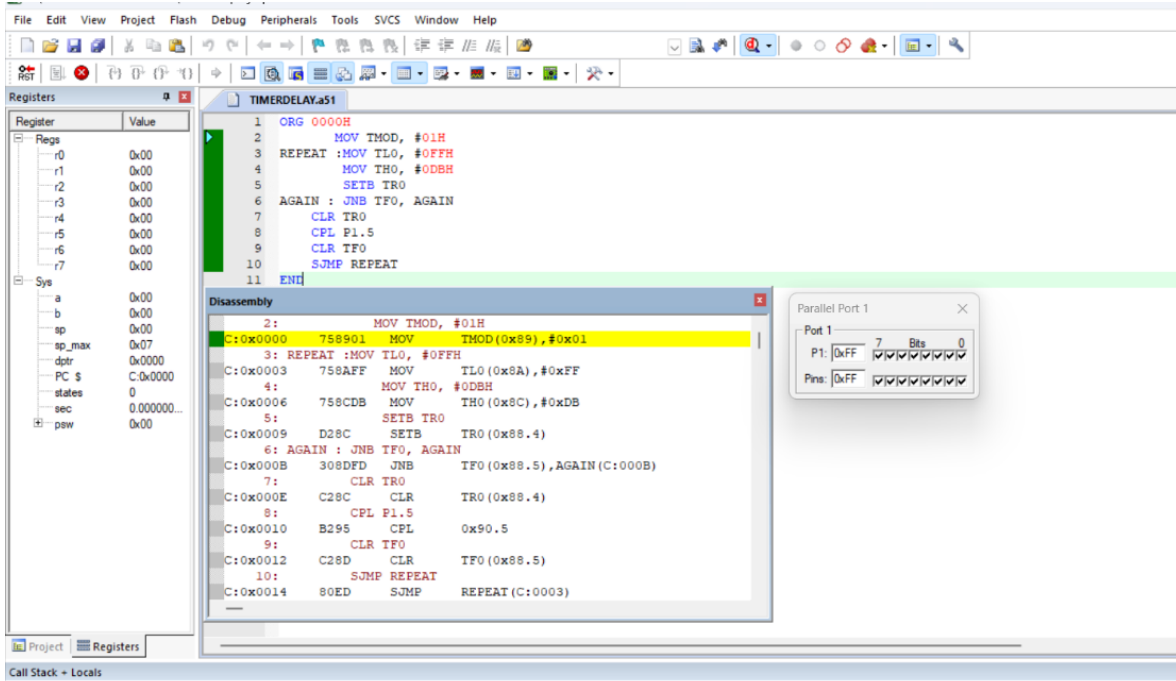


Fig: 13.9 Output Window

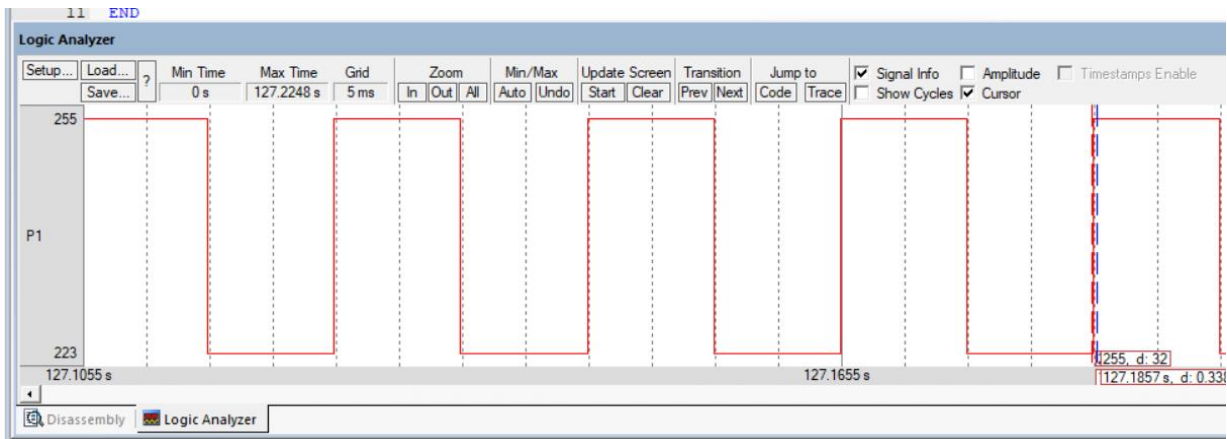


Fig 13.10: Waveform Output

Problem statement for student: Write a program to toggle Port 2 Pins with 25ms delay using Timer 1 Mode 1. Assume XTAL = 11.0592MHz

Step 1-Algorithm	Step 2-Flowchart

XII Observations for problem statement (use blank sheet provided if space not sufficient)
Draw Square wave and show TON and TOFF time

XIII Results (Output of the Program)

.....
.....

XIV Interpretation of Results (Give meaning of the above obtained results)

.....
.....

XV Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....
.....

XVI Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifies CO.

1. To get a 5 ms delay, what number should be loaded into TH, TL using mode 1. Assume XTAL= 11.0592MHz.
2. Explain Timer Mode 2 Operation.

[Space for Answers]

.....
.....
.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XVII References/Suggestions for further reading

1. http://nptel.ac.in/courses/Webcourse-contents/IIT-KANPUR/microcontrollers/micro/ui/Course_home2_8.htm
2. <http://www.circuitstoday.com/8051-timers-counters>.
3. The 8051 Microcontroller and Embedded system Using Assembly and C- Muhammad Ali Mazidi, Janice Gillispie Mazidi, Rolin D. Mckinlay- Pearson /Prentice Hall, , 2nd edition, Delhi,2008, ISBN 978-8177589030.

XVIII Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 14: Serial 8-bit data transfer on serial port

I Practical Significance

Many applications require microcontrollers to either accept the data in serial form or output the data in serial form. Serial communication is commonly used in applications such as industrial automation systems, scientific analysis and certain consumer products. This practical will help the students to develop skills to understand the concepts of serial port and serial 8 bit data transfer.

II Industry/Employer Expected Outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Develop program using timers and interrupts

IV Laboratory Learning Outcome(s)

Develop an ALP to transfer 8-bit data serially on serial port.

V Relevant Affective Domain related outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

Serial communication is the process of sending data one bit at a time, sequentially, over a communication channel. There are two methods of serial communication:

- 1) Synchronous: Transfer of block of data at a time
- 2) Asynchronous: Transfer of one byte (character) at a time

One of the major differences is that in Synchronous Transmission, the sender and receiver should have synchronized clocks before data transmission. Whereas Asynchronous Transmission does not require a clock, but it adds a parity bit to the data before transmission.

1. Synchronous data transfer.

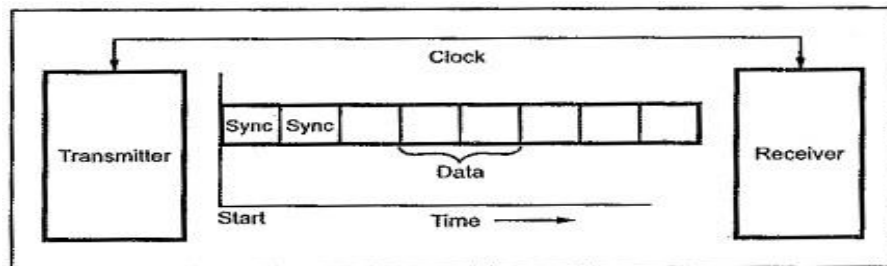


Fig. 14.1 Synchronous data transfer

2. Asynchronous data transfer.

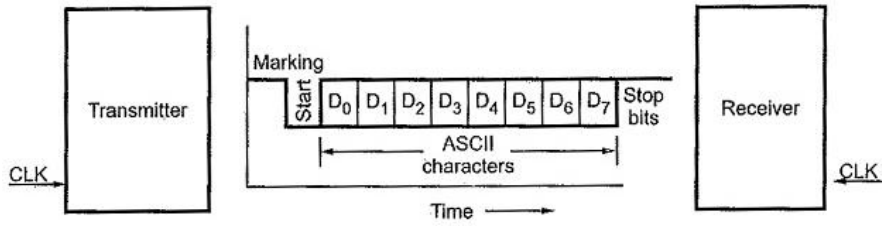


Fig. 14.2 Asynchronous data transfer

Baud Rate

In serial communication the rate at which data bits are transmitted generates a term baud rate, the baud rate is defined as bits/seconds or the changes in voltage levels/second. Standard baud rates are 1200, 2400, 4800, 9600, 14400, 19200 bits per second(bps)

Serial Port of 8051:

8051 has a built-in full duplex 8-bit UART(Universal Asynchronous Transmitter Receiver)

SCON REGISTER:

Serial control (SCON) is an 8-bit register used to control the 8051 Microcontroller's Serial Port.

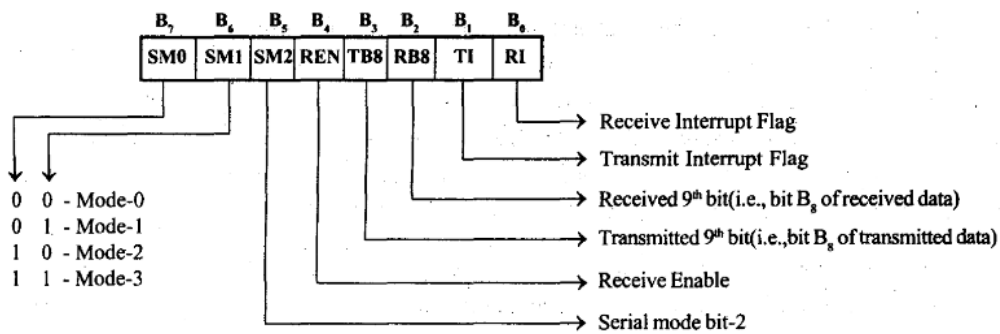


Fig. 14.3 SCON register format

SBUF Register

SBUF is a 8 bit register used in serial communication of 8051. Serial data is sent by writing to the register SBUF while data is received by reading the same register. SBUF has physically two registers, one write only and other is read only. Both registers use one address 99H

Baud Rate Generation:

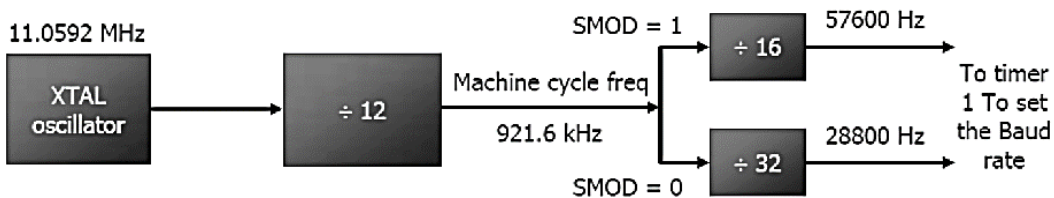


Fig. 14.4 Baud Rate generation

Timer 1, in mode 2 (8-bit, auto-reload) to generate baud rate

$$\text{Baud Rate} = \frac{2^{\text{SMOD}}}{32} \times (\text{Timer 1 Overflow Rate})$$

$$\text{Baud Rate} = \frac{2^{\text{SMOD}}}{32} \times \frac{\text{Oscillator Frequency}}{12 \times [256 - (\text{TH1})]}$$

Table No: 14.1 Values of Timer 1 Register for various Baud Rates

Baud Rate		TH1(Decimal)	TH1(Hex)
SMOD=0	SMOD=1		
9600	19200	-3	FDH
4800	9600	-6	FAH
2400	4800	-12	F4H
1200	2400	-24	E8H

Note: XTAL = 11.0592 MHZ.

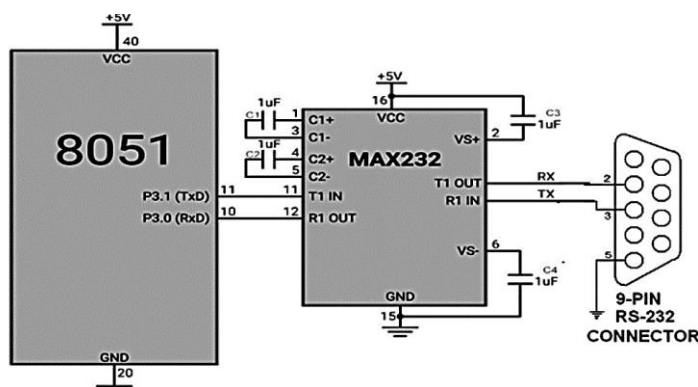


Fig. 14.5 8051 Connection to the RS232 using MAX 232

VII Required Resources/apparatus/equipment with specifications

Sr . No.	Instrument /Components	Specification	Quantity
1	Microcontroller kit	Single board systems with 8K RAM, ROM memory with battery back up,16X4,16 X2, LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross c-compiler, RS-232, USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open source IDE, simulation and program downloading software.	1 No.

VIII Precautions to be followed

- 1) Check rules / syntax of assembly language programming.

IX Procedure

Write Program

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL or INTEL and select 80c51AH 0r AT89C51.
5. Type the program in text editor and save as .asm or .a51.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window. It will display all internal registers of 8051 and their contents. The output can be observed in UART1 window.
11. Hyper Terminal, a Windows XP application, can be used to receive or transmit serial data through RS232. To open Hyper Terminal, go to Start Menu, select all programs, go to Accessories, click on Communications and select Hyper Terminal.
12. To start a new connection, go to File menu and click on new connection. The connection window opens up. Give a name to your connection and select 1st icon and click on OK. Connection property window opens here. Select Bit rate as 9600bps, Data bits 8, Parity as none, stop bit 1, Flow control none and click OK. Now the serial data can be read on hyper terminal.
13. In program, Timer1 is used with auto reload setting. The baud rate is fixed to 9600bps by loading TH1 to 0xFD. The value 0x50 is loaded in the SCON register. This will initialize the serial port in Mode1. The program continuously transmits a character (say 'A') from 8051 serial port to Serial port of the computer.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM: Write an ALP to transfer 8-bit data serially on serial port.

Step 1: Algorithm

1. Initialize TMOD register for Timer1, Mode2 and its mode.
2. Load the value in the timer register corresponding to the 9600 baud rate.
3. Load SCON with 50H value indicating mode 1, 1 stop and start bit.
4. Set the timer run bit to start Timer 1

5. Load the character 'A' in SBUF.
6. Check TI to determine whether the transmission process is completed.
7. Clear TI flag for the next character.
8. Stop

Step 2: Flowchart

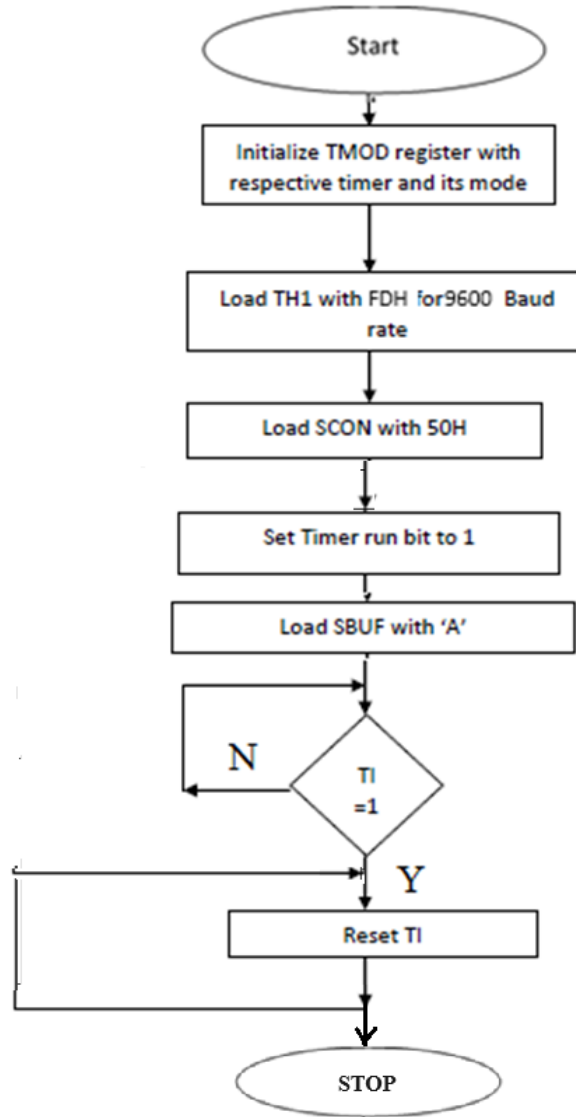


Fig 14.6 Flowchart for 8-bit data serial data transfer

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x0000	758920		MOV TMOD, #20H	;Timer 1,mode 2
C:0x0003	758DFD		MOV TH1, # -3	;9600 baud rate
C:0x0006	759850		MOV SCON, #50H	;8 bit,1 stop Bit, REN enabled
C:0x0009	D28E		SETB TR1	;Start Timer 1
C:0x000B	759941	HERE:	MOV SBUF, #'A'	;Character 'A' to be transferred

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x000E	3099FD	WAIT:	JNB TI, WAIT	;Wait for the last bit
C:0x0011	C299		CLR TI	;Clear TI for next character
C:0x0013	80F6		SJMP HERE	;Wait
			END	

Output Window

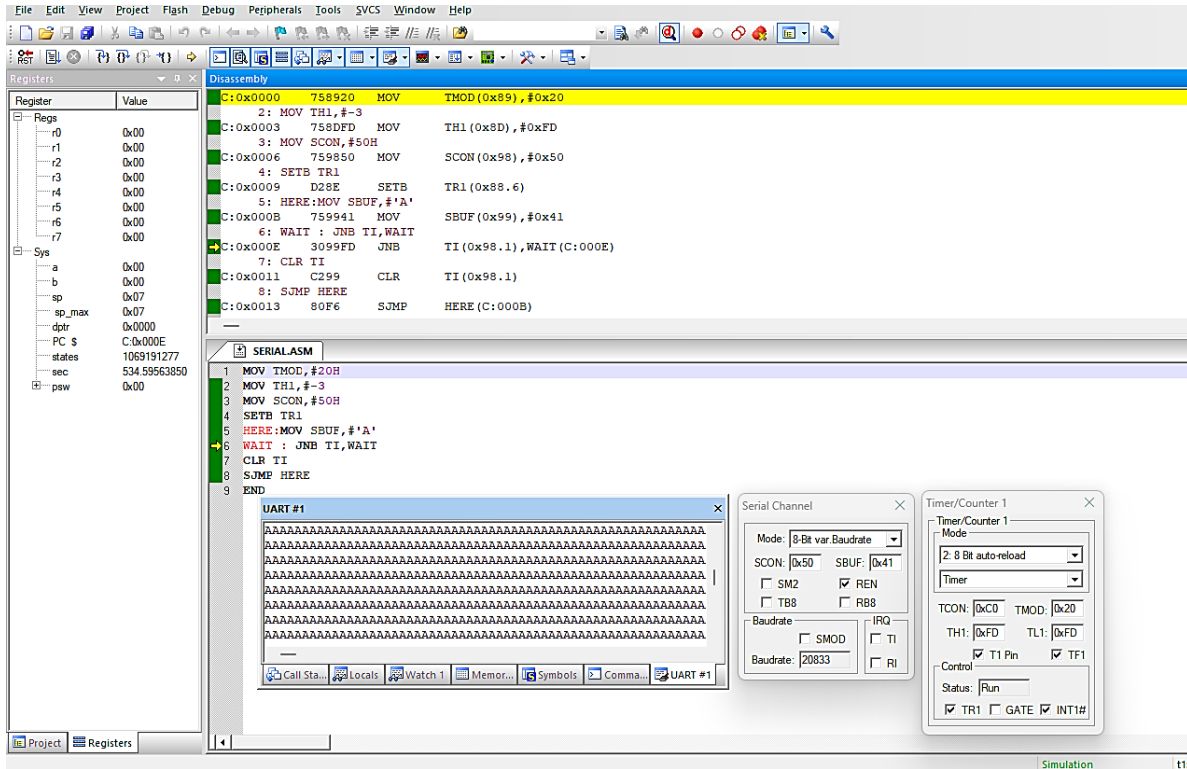


Fig 14.7 Output Window

Problem statement for student: Write a program to transfer message “MAA” at baud rate 4800 bps

Step 1-Algorithm	Step 2-Flowchart
-------------------------	-------------------------

Step 3- Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments

Memory Address	Hex Code	Label	Mnemonics	Comments

X Resources used

Sr. No.	Name of Resource	Specifications	Quantity

XI Actual Procedure Followed (use blank sheet provided if space not sufficient)

1.
2.
3.
4.
5.
6.
7.
8.

XII Observations for problem statement (use blank sheet provided if space not sufficient)
(UART window)

XIII Results (Output of the Program)

.....
.....

XIV Interpretation of Results (Give meaning of the above obtained results)

.....
.....

XV Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....
.....

XVI Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifies CO.

1. State serial communication modes of 8051 along with their baud rate.
2. State SFRs used for Serial communication in 8051.
3. State significance of TI and RI flag in Serial communication

[Space for Answers]

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

XVII References/Suggestions for further reading

1. https://embetronicx.com/tutorials/tech_devices/serial-communication-basics-tutorial-for-beginners/
2. <https://ebooks.inflibnet.ac.in/csp13/chapter/serial-port-communication/>
3. <https://www.codrey.com/embedded-systems/serial-communication-basics/>
4. <https://www.electronicwings.com/8051/8051-uart>

XVIII Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 15: LED interfacing to 8051

I Practical Significance

LED is most common semiconductor device used in many electronic system as visual indicator or signal transmission / power indication purposes. The LEDs are also used for design message display boards and traffic control signal lights etc. Interrupts are most important feature of Microcontroller. This practical will help the students to develop skills to understand the fundamental interfacing concept for 8051 microcontrollers and significance of Interrupts.

II Industry/Employer expected Outcome(s)

Maintain microcontroller- based systems

III Course Level Learning Outcome(s)

Interface memory and I/O peripherals to 8051 microcontroller.

IV Laboratory Learning Outcome

Interface LED with microcontroller and turn it “ON” with microcontroller interrupt.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

The 8051 microcontroller features four bidirectional I/O ports, each consisting of 8 bits, which can be used either as input or output. Here’s a brief overview of each port:

1. **Port 0 (P0):** This port can serve as both an input or an output port. When used as an output port, it provides open-drain outputs and requires pull-up resistors to operate as high (logic 1). Port 0 is also multiplexed with the lower order address and data bus during accesses to external memory.
2. **Port 1 (P1):** Unlike Port 0, Port 1 does not need pull-up resistors because it has internal pull-ups. When the pins are used as inputs, they are held high by these internal pull-ups and can easily be interfaced with switches.
3. **Port 2 (P2):** Similar to Port 1, this port also features internal pull-ups and serves dual purposes: it acts as a simple I/O port or, during external memory operations, it outputs the high-order address byte.
4. **Port 3 (P3):** Port 3 has multiple functions besides serving as a general I/O port. Each pin on Port 3 can also be used for special functions like interrupts, serial communication, timer inputs, and read/write operations for external memory.

Interrupt is a subroutine call that interrupts microcontrollers main operations or work and causes it to execute any other program, which is more important at the time of operation. The feature of Interrupt is very useful as it helps in case of emergency operations. An Interrupts gives us a mechanism to put on hold the ongoing operations, execute a subroutine and then again resume its original task.

1. Generally five interrupt sources are there in 8051 Microcontroller. Out of these, $\overline{\text{INT0}}$ and $\overline{\text{INT1}}$ are external interrupts that could be negative edge triggered or low level triggered. They are located on pins P3.2 and P3.3 of port 3 respectively. They are enabled or disabled using the IE register when the external interrupt flag is edge triggered, the CPU clears

interrupt flag in response to the interrupt call. When it is level triggered then the interrupt flag is cleared at high level of the interrupt signal.

2. The table shows the vector addresses for the interrupts:

Interrupt	Flag	Vector
System reset	RST	0000H
External interrupt 0	IE0	0003H
Timer 0	TF0	000BH
External interrupt 1	IE1	0013H
Timer 1	TF1	001BH
Serial port	RI or TI	0023H

Light emitting diodes: LEDs are the most commonly used components in many applications. It has two terminals positive and negative as shown in the figure:

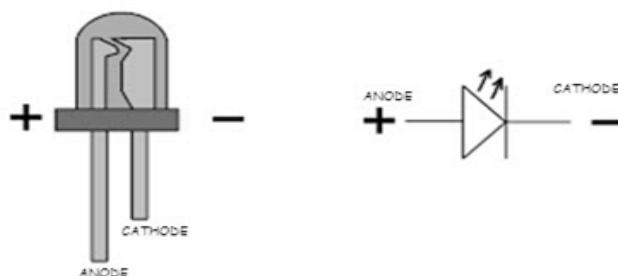


Fig 15.1 LED diagram

Commonly used LEDs will have voltage drop of 1.9v to 2.1v and current of 15mA (Typically) or 20mA (high brightness) to glow at full intensity. This is applied through the output pin of the microcontroller.

VII Actual Circuit diagram used in laboratory with related equipment rating

a) Sample Circuit diagram

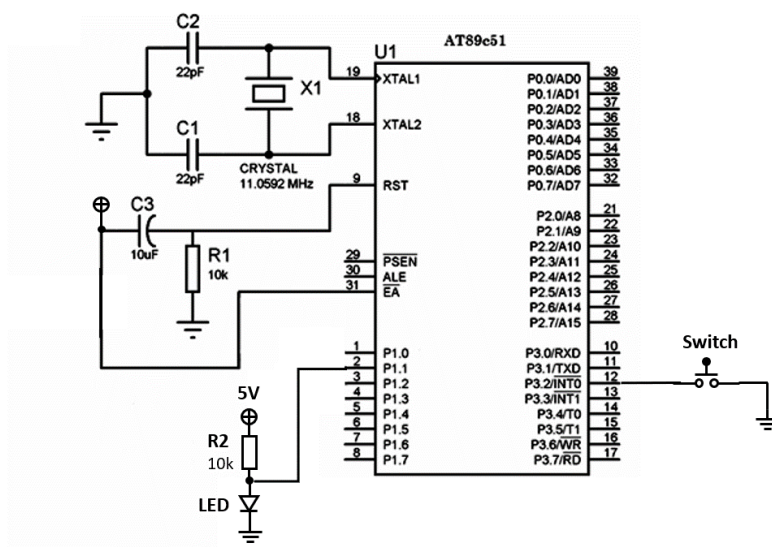


Fig 15.2 8051 connection to LED and switch

b) Practical setup

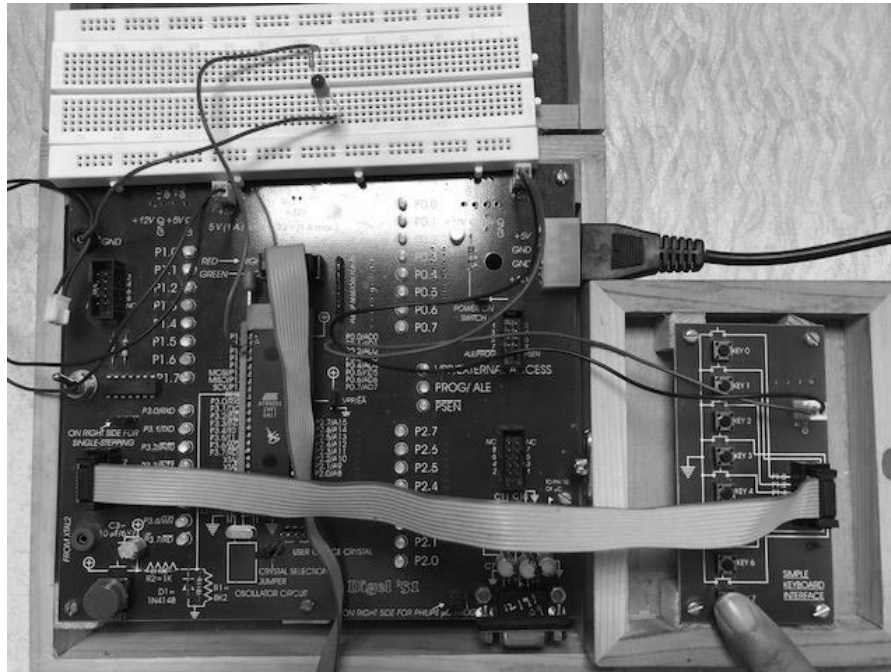


Fig 15.3 Practical Setup

c) Simulation diagram

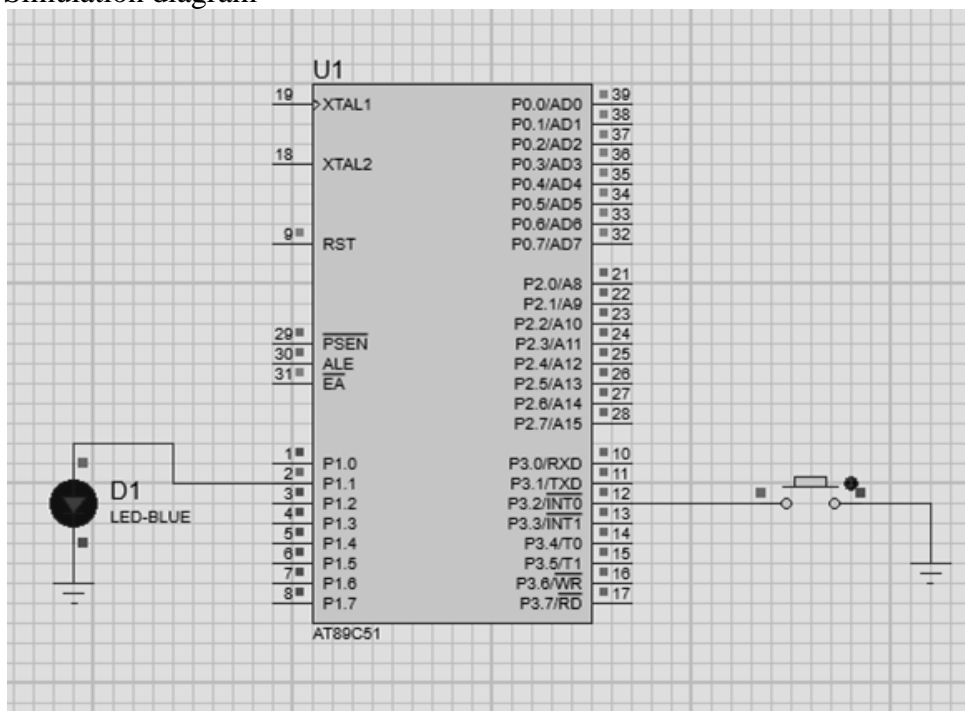


Fig 15. 4 Simulation diagram

d) Actual circuit used in Laboratory

e) Actual Experimental set up used in laboratory

VIII Resources Required

Sr. No.	Instrument /Components	Specification	Quantity
1.	Microcontroller kit	Single board system with 8K RAM, ROM memory with battery backup, 16X4, 16X2 LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross-c-compiler, RS-232, USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open-source IDE, simulation and program downloading software.	1 No.

IX Precautions to be followed

1. Use always current limiting resistor before LED connected to microcontroller

X Procedure

1. Write algorithm for given problem.
2. Draw flowchart for the same.
3. Develop assembly program using Integrated Development Environment (Keil IDE) or any other relevant software tool.
4. Debug program on IDE.
5. Execute program on IDE.
6. Create hex file for the program.
7. Download hex code in EPROM/Flash memory of microcontroller.
8. Interface LED to microcontroller as per circuit diagram shown in fig.
9. Observe the LED to glow when external interrupt occurs on P3.2.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

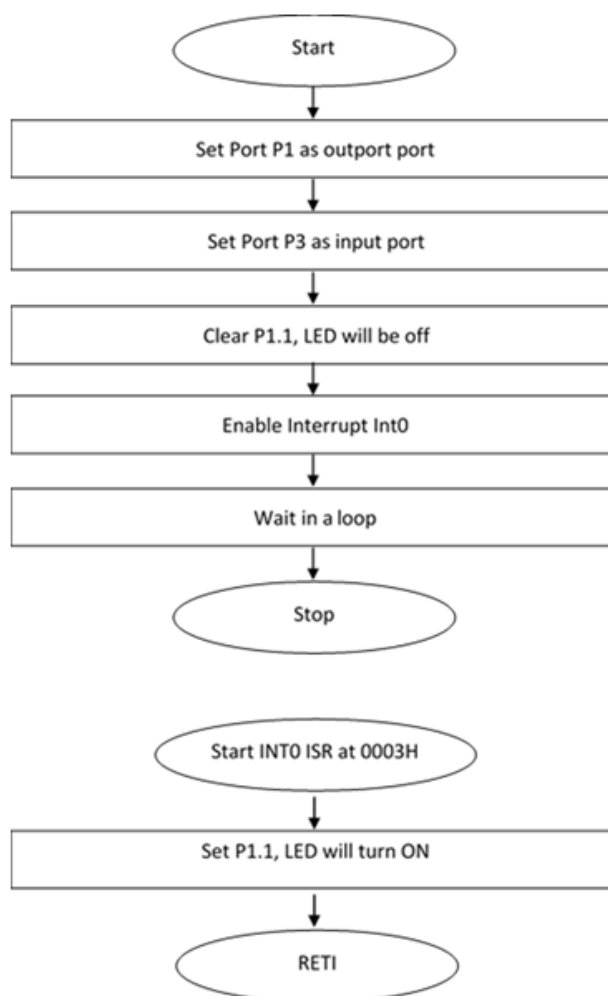
SAMPLE PROGRAM 1: Interface LED with microcontroller and turn it ON with microcontroller interrupt.

Step 1: Algorithm**Main Program**

1. Initialize port P1 as output.
2. Initialize port P3 as input.
3. Clear pin P1.1
4. Enable interrupt INT0.
5. Wait for Interrupt.

INT0 ISR

1. Set pin P1.1
2. Return from ISR

Step 2: Flowchart**Fig 15 .6 Flowchart to turn LED ON with microcontroller interrupt.****Step 3: Assembly Language Program**

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0x0000	020006		LJMP START	
			ORG 0003H	Interrupt service routine for INTO
C:0x0003	D291		SETB P1.1	
C:0x0005	32		RETI	
C:0x0006	759000	START:	MOV P1, #00H	Main program for initialization
C:0x0009	75B0FF		MOV P3, #0FFH	
C:0x000C	75A881		MOV IE, #81H	Enable hardware interrupt INTO
C:0x000F	80FE	HERE:	SJMP HERE	
			END	

Problem statement for student Interface two LEDs with microcontroller and turn them ON with microcontroller interrupts.

Step 1: Algorithm	Step 2: Flowchart

Step 3- Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments

XI Resources Used

S. No.	Instrument /Components	Specification	Quantity
1.			
2.			
3.			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

1.
2.
3.
4.
5.
6.
7.
8.

XIII Observations (use blank sheet provided if space not sufficient)

LED will become _____ (ON/OFF) after occurrence of _____ (INT0/INT1)
Interrupt

XIV Result (Output of the Program)

.....
.....

XV Interpretation of Results (Give meaning of the above obtained results)

.....
.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results)

.....
.....

XVIII References / Suggestions for further reading

1. <https://vision-pi.net/common-cathode-vs-common-anode-led/>
2. <https://www.fypsolutions.com/assembly-language/8051-8052/led-blink-8051-assembly/>
3. <https://www.elprocus.com/led-interfacing-with-8051-microcontroller/>

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related (15 Marks)		60% (15)
7	Coding and Debugging ability	30%
8	Making connections of hardware	20%
9	Follow ethical practices.	10%
Product related (10 Marks)		40%(10)
10	Correctness of algorithm/ Flow chart	20%
11	Relevance of output of the problem definition.	15%
12	Timely Submission of report, Answer to sample questions.	05%
TOTAL		100% (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 16: Generating Pulse and Square wave using timer delay

I Practical Significance

The input/output (I/O) ports allow the microcontroller to connect to external devices and peripherals. In 8051 timers are used to generate delays or as counters to count events happening outside the microcontroller. In time required applications two available 16-bit timers are operated in different modes to generate specific delay. This practical will help the students to develop skills to program timers and generate delays for generating square wave on I/O port pin.

II Industry/Employer Expected Outcome(s)

Maintain microcontroller based systems.

III Course Level Learning Outcome(s)

Interface the memory and I/O peripherals to 8051 microcontroller

IV Laboratory Learning Outcome(s)

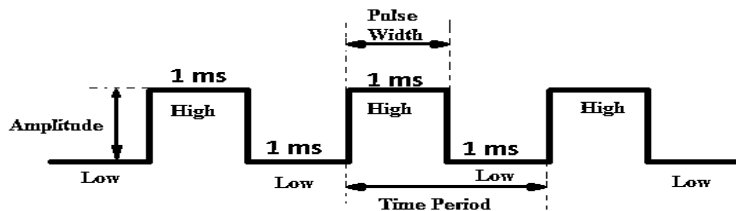
Develop an ALP to generate pulse and square wave by using timer delay.

V Relevant Affective Domain related outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

Square wave of any frequency (limited by the controller specifications) can be generated using the 8051 timer. Square wave generation requires a port pin to output logic high ('1') and logic low ('0') level alternately with delay in between. The delay time is equal to half of time period of square wave.



$$\text{Time Period} = (1+1)\text{ms} = 2\text{ms}$$

$$\begin{aligned} \text{Duty Cycle} &= \frac{\text{Pulse width}}{\text{Time Period}} \times 100 \\ &= \frac{1}{2} \times 100 \\ &= 50\% \end{aligned}$$

Fig 16.1 Square Wave calculation

$$\text{Frequency} = 1/\text{Time Period} = 1/2\text{ms} = 0.5\text{KHZ} = 500\text{HZ}$$

Port pin can be toggled by using instruction CPL bit-address and /or by setting and resetting the port pin by using instruction SETB bit-address and CLR bit address respectively.

Square wave can be generated on port pin of microcontroller 8051 with precise TON and TOFF time by using Timers to generate delay. The frequency of the square wave generated depends on the count value loaded in timer registers and frequency of oscillator. 8051 has two timers, Timer 0 and Timer 1. Both are 16 bit up-counter. Timers operate in four modes. Timer count Increments at the rate of $(F_{osc}/12)$, which is 1 machine cycle. After roll-over timer overflow flag is set

To generate a Pulse / square wave of a particular pulse width , the delay time equal to pulse width has to be generated using Timer.

Generation of Pulse width delay using Timer Mode 1:

$$\begin{aligned} \text{Required time delay} &= (12/F_{osc}) \times \text{number of increments (N)} \\ 1 \text{ ms} &= (12/ 11.0592\text{MHZ}) \times \text{number of increments (N)} \\ 1 \text{ ms} &= 1.085 \text{ usec.} \times N \\ N &= 1\text{ms} / 1.085\text{usec.} \\ N &= 921.65 = 922 \end{aligned}$$

Using Timer MODE 1,
 COUNT = $2^{16} - N$
 COUNT = 65536 – 922
 COUNT = $(64614)_{10} = \text{FC66H}$
 Therefore THx = FCH
 TLx = 66H

Generation of 2 KHZ Square wave of 50 % duty cycle

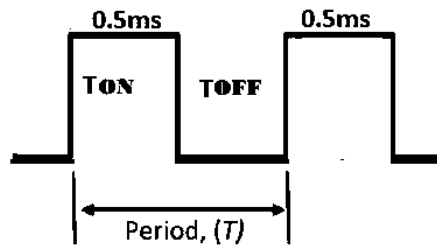


Fig 16.2 : 50% Duty cycle Square wave

Frequency = 2 KHZ
 Time period T= 1/2 KHZ = 0.5 ms
 Required time delay = TON = TOFF = T /2 = 0.5 ms / 2 = 0.25ms

VII Practical Circuit diagram:

a) Sample Circuit diagram

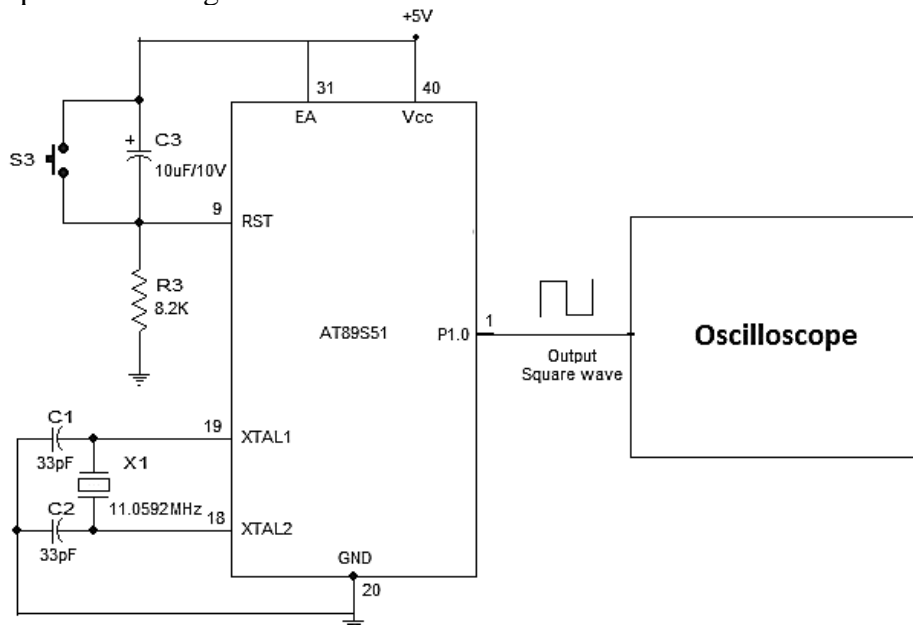


Fig 16.3 : 8051 connection to CRO

b) Simulation diagram

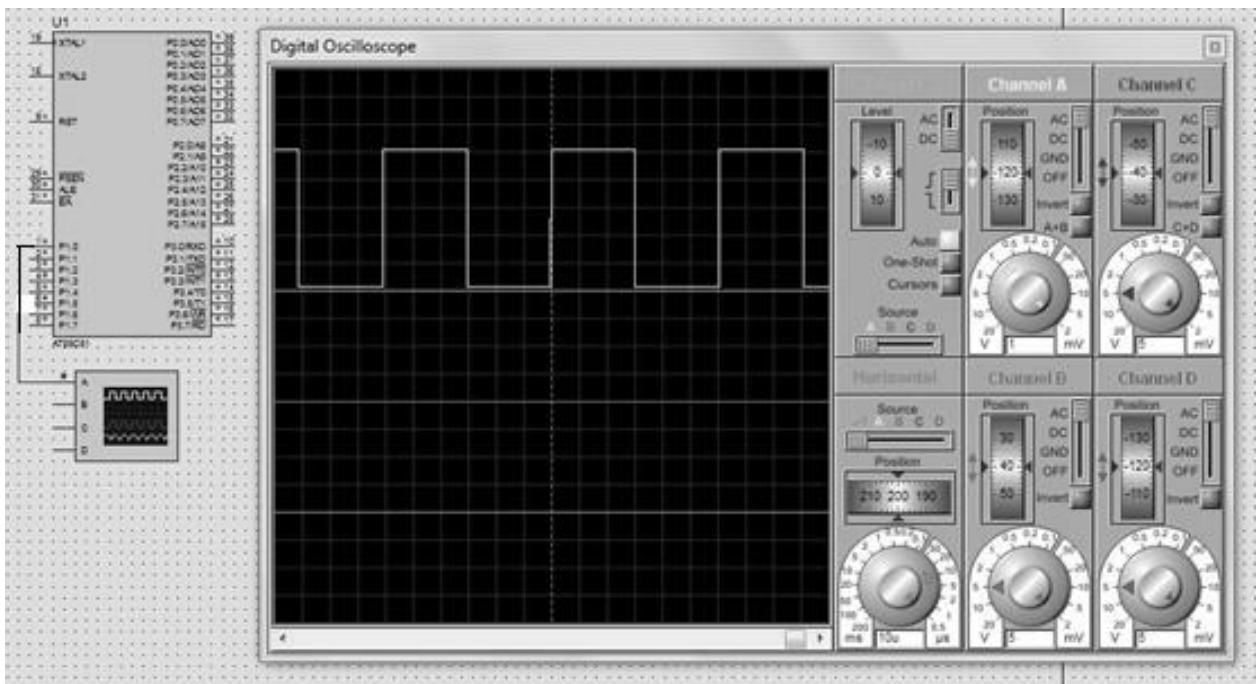


Fig 16.4 Simulation diagram

c) Actual circuit used in laboratory

VIII Required Resources/apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1.	Microcontroller kit	Single board system with 8K RAM,ROM memory with battery backup,16X4,16X2LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross C-compiler,RS-232,USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open source IDE, simulation and program downloading software.	1 No.
3.	CRO	Bandwidth AC 10Hz ~ 20MHz (-3dB). DC ~ 20MHz (-3dB), X10 Probe	1 No.

IX Precautions to be followed

1) Check rules / syntax of assembly language programming.

X Procedure

1. Write algorithm for given problem.
2. Draw flowchart for the same.
3. Develop assembly program using Integrated Development Environment (Keil IDE) or any other relevant software tool.
4. Debug program on IDE.
5. Execute program on IDE.

6. Create hex file for the program.
7. Download hex code in EPROM/Flash memory of microcontroller.
8. Connect CRO probe to port pin and observe waveform.
9. Measure ON time and OFF time on CRO and draw the same in observation Table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM: To generate Pulse with pulse width of 1ms on P1.0 using Timer 0 Mode1. Assume XTAL = 11.0592MHz

Step 1: Algorithm

1. Load the TMOD register with value 01H, indicating Timer 0, Mode1
2. Clear Port pin P1.0
3. Call 1ms delay subroutine
4. Load registers TL0 and TH0 with initial count values for 1ms delay
5. Start the Timer 0.
6. Keep monitoring the timer flag (TF0) with the “JNB TF0, target” instruction to see if it is raised. Get out of the loop when TF0 becomes high.
7. Stop Timer0
8. Clear the TF0 flag for the next round.
9. Set Port pin P1.0
10. Go back to Step 3.

Step 2: Flowchart

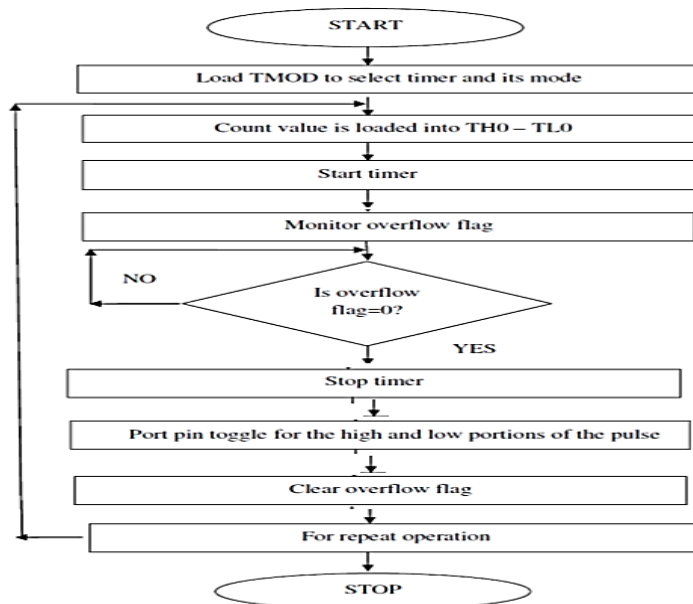


Fig 16.3 Flowchart for Pulse generation

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0X0000	758901		MOV TMOD ,#01H	;Timer 0, mode 1
C:0X0003	C290	UP:	CLR P1.0	; Clear port pin P1.0
C:0X0005	110D		ACALL DELAY	; call 1ms delay subroutine
C:0X0007	D290		SETB P1.0	; Set port pin P1.0
C:0X0009	110D		ACALL DELAY	; call 1ms delay subroutine
C:0X000B	80F6		SJMP UP	; Repeat
C:0X000D	758A66	DELAY:	MOV TL0,#66H	;TL0=66H
C:0X0010	758CFC		MOV TH0,#0FCH	;TH0=FCH
C:0X0013	D28C		SETB TR0	;start Timer 0
C:0X0015	308DFD	WAIT:	JNB TF0, WAIT	;monitor Timer 0 overflow flag until it rolls over
C:0X0018	C28C		CLR TR0	;stop Timer 0
C:0X001A	C28D		CLR TF0	;clear Timer 0 overflow flag
C:0X001C	22		RET	;Return to main program
			END	

Problem statement for student: Write a program to generate square wave of frequency 2KHZ on port pin P2.5 Using Timer 1, Mode 1. Assume XTAL = 11.0592MHz

Step 1-Algorithm	Step 2-Flowchart

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

1.
2.
3.
4.
5.
6.
7.
8.

XIII Observations for problem statement (use blank sheet provided if space not sufficient)

Draw Square wave and show TON and TOFF time

XIV Results (Output of the Program)

.....
.....

XV Interpretation of Results (Give meaning of the above obtained results)

.....
.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....
.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XVII References/Suggestions for further reading

1. https://www.nbcafe.in/ports-in-8051-microcontroller/#google_vignette
2. https://econtent.msbt.edu.in/econtent/econtent_home.php
3. <https://www.circuitstoday.com/delay-using-8051-timer>

XVIII Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 17: LED matrix interfacing to 8051

I Practical Significance

An LED matrix keyboard is a specialized input device that incorporates an LED matrix to enhance visual feedback and aesthetic appeal. LED matrix displays are used as stadium displays, decorative displays and as visual signals to human eye, to convey a message or meaning. LED matrix displays are interfaced with microcontroller I/O port to display characters and different patterns. This practical will help the students to develop skills to interface LED matrix display to microcontroller and display various pattern.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Interface memory and I/O peripherals to 8051 microcontroller.

IV Laboratory Learning Outcome(s)

Interface 4x4 matrix with 8051 to display various patterns.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

In LED dot matrix display the LEDs are connected at the column and row intersections of the matrix. LEDs in the same row are connected together and LEDs in the column are connected together. Transistors are act as switches and used to control LEDs in the matrix

LED (Light Emitting Diode)

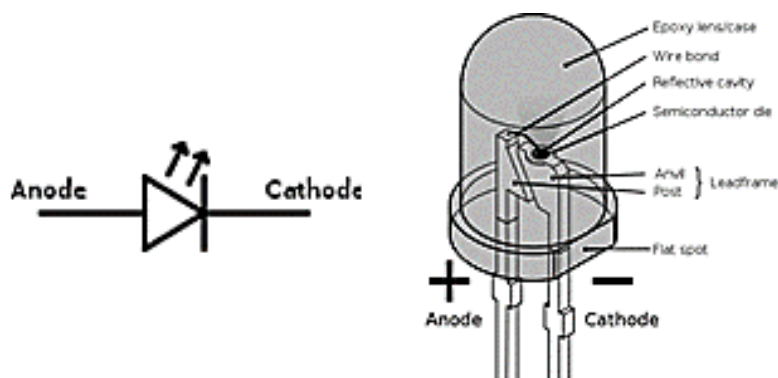


Fig 17.1 LED Symbol and Construction

Specifications: LED

1. Current: 20 mA
2. Voltage drop: 1.9 to 2.1 V
3. Power dissipation: 40 mW
4. Color: RED

Operation: Voltage +5V to anode with respect to cathode ground LED will turn ON.

Addressing individual LEDs :User can turn ON an individual LED by setting its row and column pins to the proper logic 1.

For example, referring to figure 17.2 LED matrix the switch in column 2 is closed which ties the anodes of all of the LEDs in that column to positive voltage and on the left the switch in row 1 is closed causing a ground level to be applied to the cathode of all of the LEDs in that row. The LED at the intersection of column 2 and row 1 is forward biased and turns on.

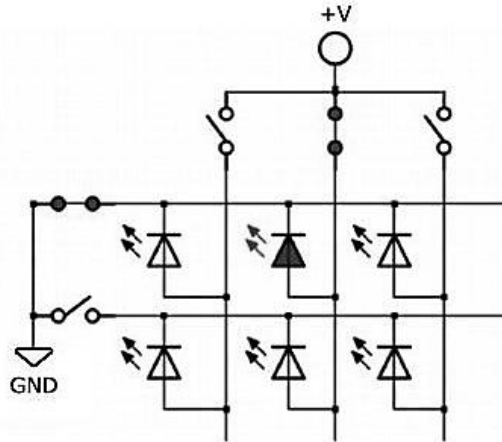


Fig 17.2 Addressing individual LED: 2nd column and 1st row

VII Actual Circuit Diagram used in laboratory

a) Sample Circuit Diagram

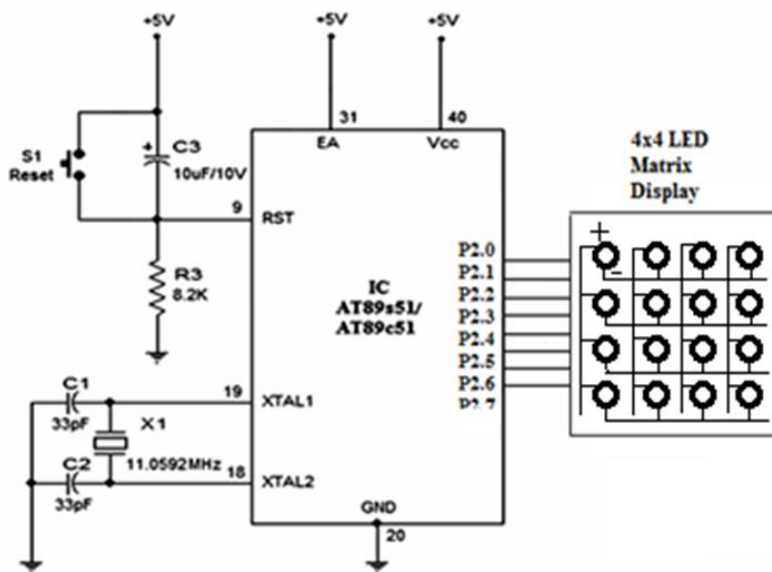


Fig 17.3 Sample Circuit Diagram

b) Actual Circuit Diagram used:**VIII Resources Required**

Sr. No.	Instrument /Components	Specification	Quantity
1	Microcontroller kit	Single board system with 8K RAM,ROM memory with battery backup,16X4,16X2LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross c-compiler,RS-232,USB, interfacing facility with built in power supply.	1 No.
2	Desktop PC	Loaded with open source IDE, simulation and program downloading software.	1 No.
3	4X4 LED matrix	Suitable to interface with 8051 trainer kit	1 No

IX Precautions to be Followed

1. Check rules / syntax of assembly programming.

X Procedure**Write Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.
Run, Debug the Program
8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write program to display various patterns on 4x4 LED matrix.

Step 1: Algorithm

1. Make port P2 used to interface 4x4 LED matrix as an output port.
2. Send F1H value to the port to turn ON first column all LED.
3. Add delay.
4. Send 1FH value to the port to display first row all LED.
5. Add delay.
6. Send 11H value to the port to display first column first row LED.
7. Add delay.
8. Send 88H value to the port to display fourth column fourth row LED.
9. Add delay.
10. For repeat operation go to step 2.

Step 2: Flowchart

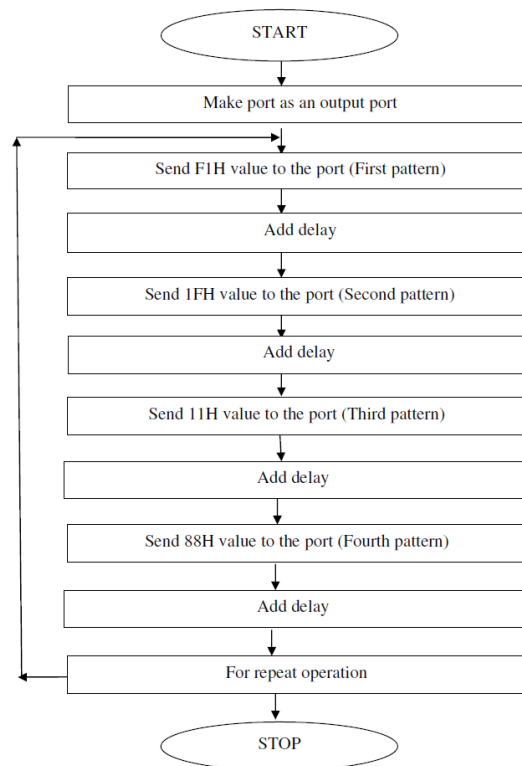


Fig . 11.2 Flowchart for 4x4 LED matrix

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0x0000	75A000		MOV P2, #00H	Make port as output
C:0x0003	75A0F1		MOV P2, #0F1H	Send value to LED matrix
C:0x0006	1119	RPT:	ACALL DELAY	Add delay
C:0x0008	75A01F		MOV P2, #1FH	Send value to LED matrix
C:0x000B	1119		ACALL DELAY	Add delay
C:0x000D	75A011		MOV P2, #11H	Send value to LED matrix
C:0x0010	1119		ACALL DELAY	Add delay
C:0x0012	75A088		MOV P2, #88H	Send value to LED matrix
C:0x0015	1119		ACALL DELAY	Add delay
C:0x0017	80EA		SJMP RPT	
C:0x0019	7A0A		MOV R2, #10	Delay subroutine
C:0x001B	7B64	DELAY:	MOV R3, #100	
C:0x001D	7CC8	HERE2:	MOV R4, #200	
C:0x001F	DCFE	HERE1:	DJNZ R4, HERE	
C:0x0021	BDF6	HERE:	DJNZ R3, HERE1	
C:0x0023	DAF6		DJNZ R2, HERE2	
C:0x0025	22		RET	
			END	

XI Resources Used:

S. No.	Instrument /Components	Specification	Quantity
1.			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XIII Observations for sample program (use blank sheet provided if space not sufficient)

HEX Value	LED status (ON/OFF)	
	Column	Row
F5H		
2FH		
17H		
90H		

XIV Results (Output of the Program)

.....

.....

.....

.....

XV Interpretation of Results (Give meaning of the above obtained results)

.....

.....

.....

.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....

.....

.....

.....

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. Give the difference between Common Anode and Common Cathode Display.
2. Specify the power requirement for 4x4 LED Matrix.
3. List the applications of 4 x 4 matrix keyboard.

[Space for Answers]

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XVIII References / Suggestions for further reading

1. <https://www.refreshnotes.com/2016/04/8051-program-exchange-block-of-data.html>
2. <https://www.tutorialspoint.com/program-branch-group-in-8051>
3. <https://www.codesexplorer.com/2016/12/8051-alp-to-move-data-from-internal-to-external.html>

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40% (10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 % (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 18: Seven Segment Display interface for displaying decimal numbers

I Practical Significance

Seven segment display is a output display device used to display information in the form of decimal numbers from 0 to 9 and in some cases, basic characters also. It is widely used in digital clocks, basic calculators, electronic meters, and other electronic devices that display numerical information. This practical will help the students to develop skills to interface 7-segment display to microcontroller and display decimal numbers.

II Industry/Employer Expected Outcome(s)

Maintain microcontroller based systems.

III Course Level Learning Outcome(s)

Interface the memory and I/O peripherals to 8051 microcontroller

IV Laboratory Learning Outcome(s)

Interface 7-segment display to display the decimal number from 0 to 9.

V Relevant Affective Domain related outcome(s)

1. Follow safe practices.
2. Maintain tools and equipment.
3. Follow ethical practices.

VI Relevant Theoretical Background

Seven segment displays are of two types, i) Common Cathode Display ii) Common Anode Display

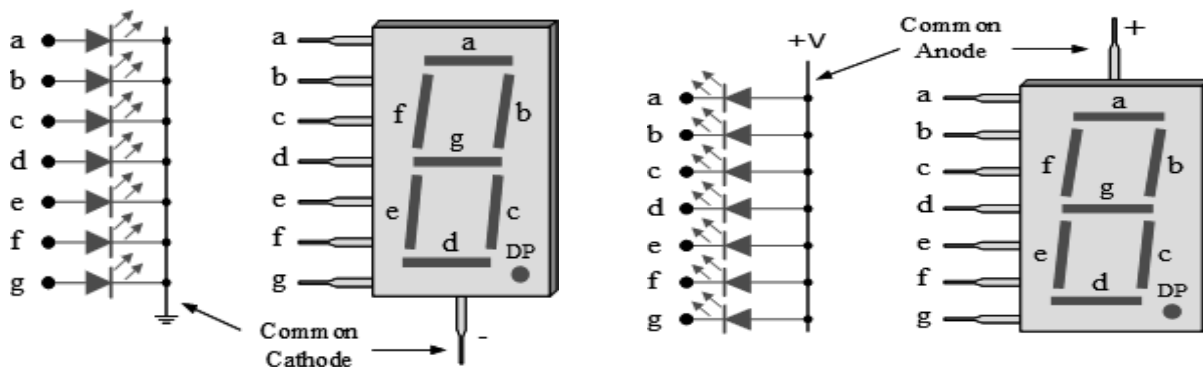


Fig 18.1 Seven segment display types

Common Cathode: In common cathode type, the cathodes of all LEDs are tied together to a single terminal which is usually labeled as ‘com’ and the anode of all LEDs are left alone as individual pins labeled as a, b, c, d, e, f, g and h (or dot). To glow Common Cathode Segment LED, common terminal is grounded and Logic 1 is applied on segment pin.

Common Anode: In common anode type, the anode of all LEDs are tied together as a single terminal and cathodes are left alone as individual pins. Common Anode is tied to logic 1 and to glow segment Led logic 0 is applied on segment Pin

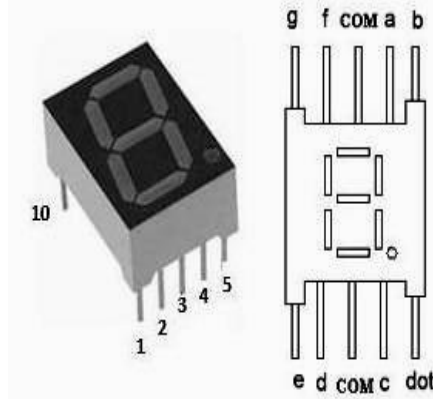


Fig 18.2 Seven segment display and pin configuration

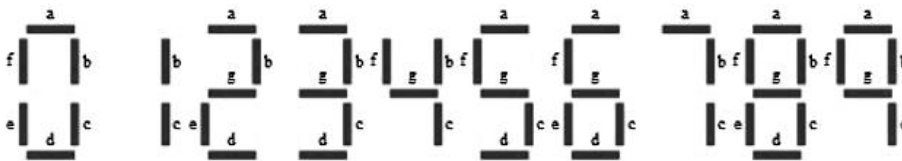


Fig 18.3 Seven segment display number pattern

Driving Pattern for Digit: Digit drive pattern of a seven segment LED display is simply the different logic combinations of its terminals ‘a’ to ‘h’ in order to display different digits and characters. The common digit drive patterns (0 to 9) of a seven-segment display are shown in the table below.

Common Cathode Pattern:

Table 18.1: Common Cathode Pattern Table

Digit	Dp	g	f	e	d	c	b	a	HEX CODE
Port	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
0	0	0	1	1	1	1	1	1	3FH
1	0	0	0	0	0	1	1	0	06H
2	0	1	0	1	1	0	1	1	5BH
3	0	1	0	0	1	1	1	1	4FH
4	0	1	1	0	0	1	1	0	66H
5	0	1	1	0	1	1	0	1	6DH
6	0	1	1	1	1	1	0	1	7DH
7	0	0	0	0	0	1	1	1	07H
8	0	1	1	1	1	1	1	1	7FH
9	0	1	1	0	1	1	1	1	6FH

Common Anode Pattern:

Table 18.2: Common Anode Pattern Table

Digit	Dp	g	f	e	d	c	b	a	HEX CODE
Port	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.0	
0	1	1	0	0	0	0	0	0	C0H
1	1	1	1	1	1	0	0	1	F9H
2	1	0	1	0	0	1	0	0	A4H
3	1	0	1	1	0	0	0	0	B0H
4	1	0	0	1	1	0	0	1	99H
5	1	0	0	1	0	0	1	0	92H
6	1	0	0	0	0	0	1	0	82H
7	1	1	1	1	1	0	0	0	F8H
8	1	0	0	0	0	0	0	0	80H
9	1	0	0	1	0	0	0	0	90H

Note: in above patterns decimal point is considered OFF

VII Practical Circuit diagram:

a) Sample Circuit diagram

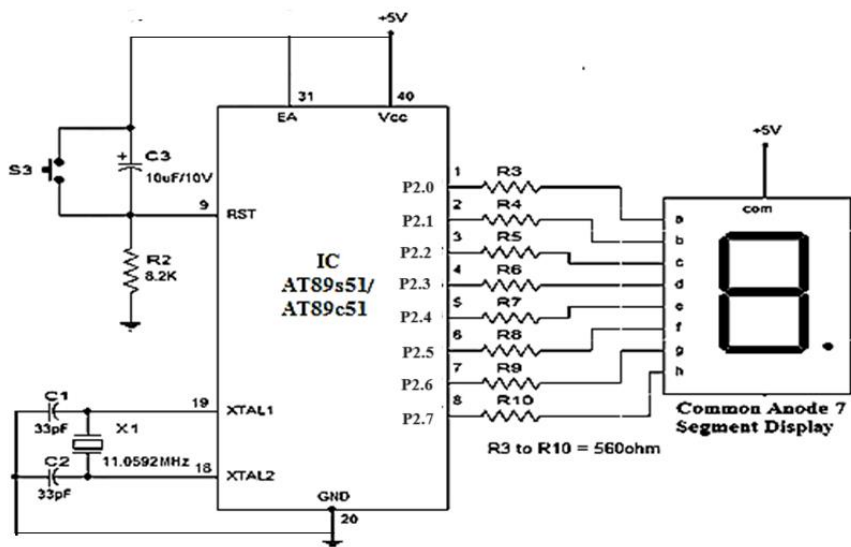


Fig 18.4 8051 connection to Common Anode seven segment display

b) Practical Setup

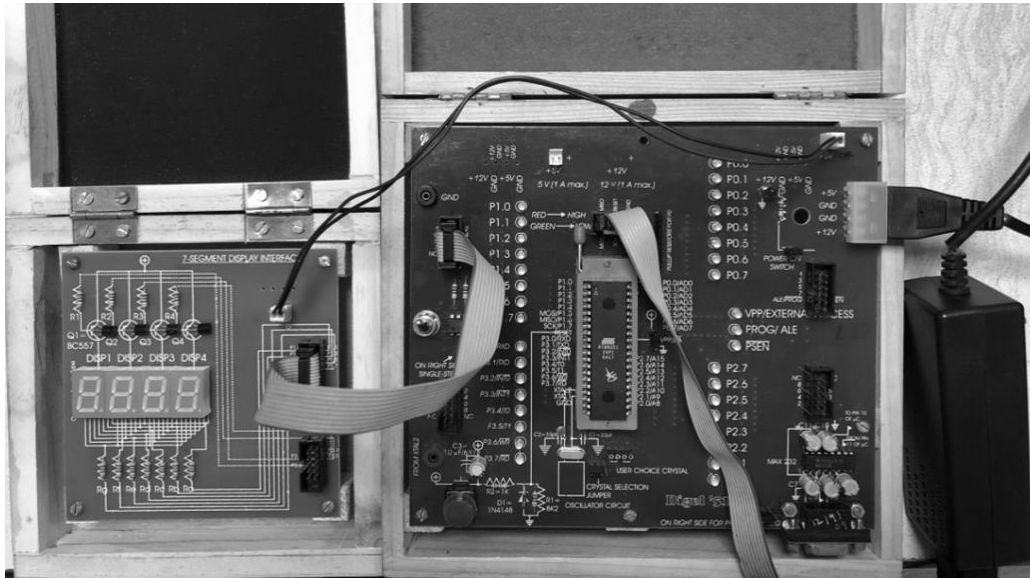


Fig 18.5 Practical setup

c) Simulation diagram

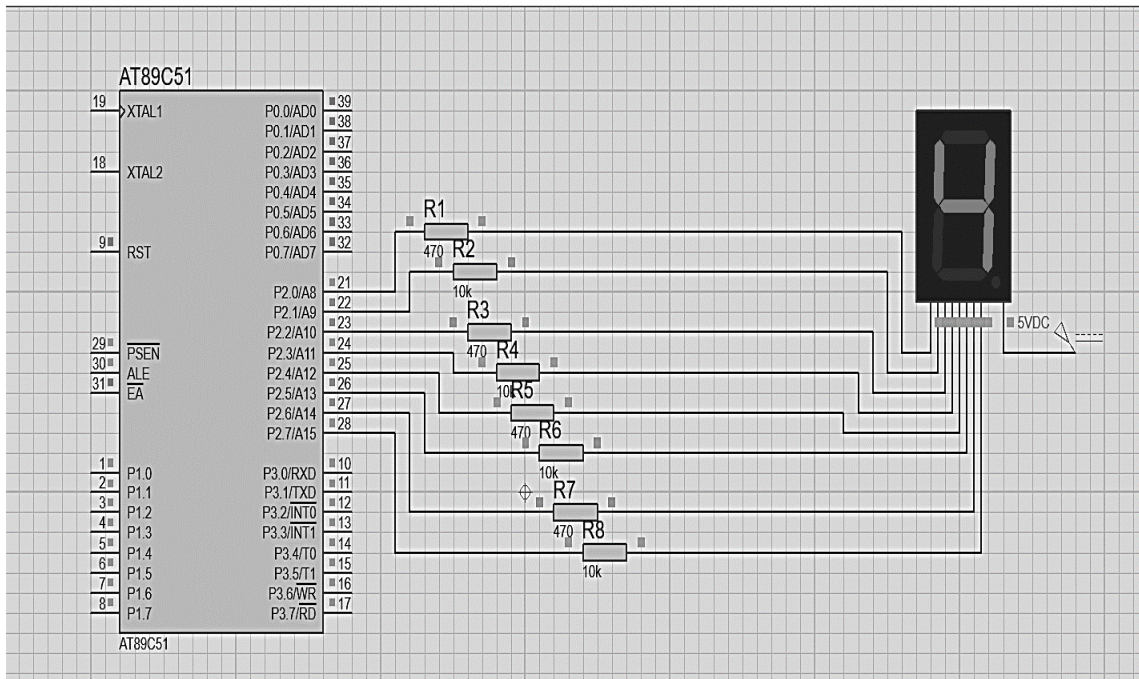


Fig 18.6 Simulation diagram

d) Actual circuit used in Laboratory

e) Actual Experimental set up used in laboratory

VIII Required Resources/apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1.	Microcontroller kit	Single board system with 8K RAM, ROM memory with battery backup, 16X4, 16X2 LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross C-compiler, RS-232, USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open-source IDE, simulation and program downloading software.	1 No.
3.	Seven Segment LED Display	0.56 in 1-digit Red, common anode/common cathode display.	1 No.

IX Precautions to be followed

- 1) Always use current limiting resistor before interfacing 7-segment display to microcontroller.
- 2) For safe operation use seven segment displays at 25° temperature.
- 3) Check rules / syntax of assembly language programming.

X Procedure

1. Write algorithm for given problem.
2. Draw flowchart.
3. Develop assembly program using Integrated Development Environment (Keil IDE) or any other relevant software tool.
4. Debug program on IDE.
5. Execute program on IDE.
6. Create hex file.
7. Download hex code in EPROM/Flash memory of microcontroller.
8. Interface Common Anode type 7 segment display to microcontroller as per circuit diagram shown in fig 18.4.
9. Observe and draw the display of numbers on 7-segment display.
10. Record the hex value in observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write a program to display decimal no 0 to 9.

Step 1: Algorithm

1. Make the Port P2 as output port.
2. Set counter register R2 =10 for 0 to 9 digits.
3. Load DPTR with memory address where table is stored.
4. Clear Accumulator.
5. Read stored hex code of decimal digit from memory into Accumulator.
6. Send code to output port where 7-segment display is connected.
7. Increment memory pointer i.e., DPTR.
8. Decrement the counter register R2 and compare with 0 is counter =0? NO- go to step 4 to send next digit code.
9. For repeat operation go to step 2.
10. Stop

Step 2: Flowchart

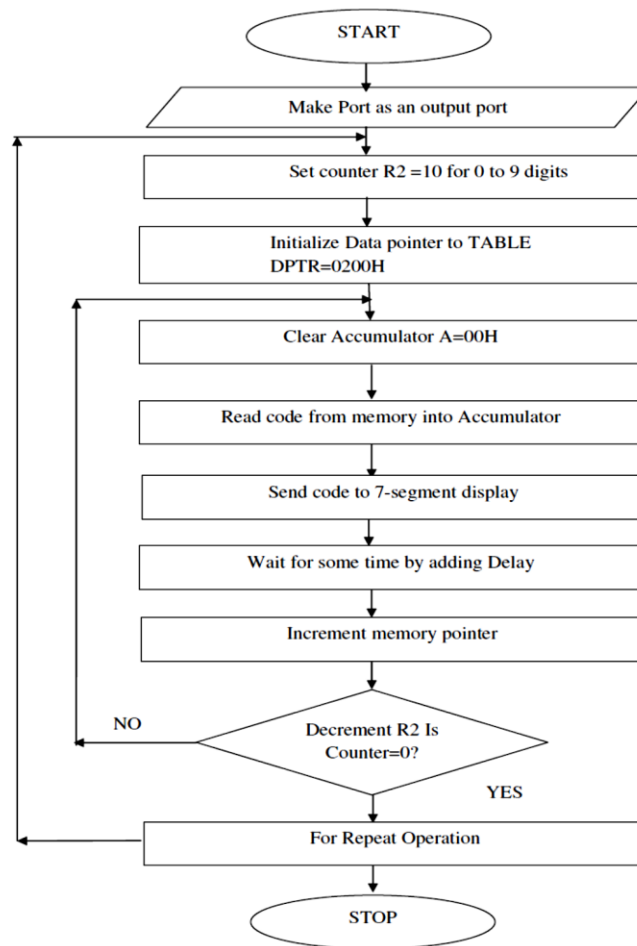


Fig 18.7 Flowchart to display decimal no 0 to 9

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0x0000	75A000		MOV P2,#00H	;Make Port2 as output port
C:0x0003	7A0A	REPEAT:	MOV R2,#10	;Set register as counter of 10 bytes
C:0x0005	900300		MOV DPTR,#TABLE	;Load address of memory into Data pointer
C:0x0008	E4	UP:	CLR A	;Clear accumulator
C:0x0009	93		MOVC A,@A+DPTR	;Read hex code from memory into accumulator
C:0x000A	F5A0		MOV P2,A	;Send hex code to port2
C:0x000C	1113		ACALL DELAY	
C:0x000E	A3		INC DPTR	;Increment memory pointer to read next digit hex code
C:0x000F	DAF7		DJNZ R2,UP	; Decrement counter & jump if not equal to zero to label UP.
C:0x0011	80F0		SJMP REPEAT	;Repeat loop
C:0x0013	7B19	DELAY:	MOV R3, #25	;Delay Subroutine
C:0x0015	7C64	L3:	MOV R4,#100	
C:0x0017	7D64	L2:	MOV R5,#100	
C:0x0019	DDFE	L1:	DJNZ R5,L1	
C:0x001B	DCFA		DJNZ R4,L2	
C:0x001D	DBF6		DJNZ R3,L3	
C:0x001F	22		RET	
			ORG 0300H	
		TABLE:	DB 0C0H, 0F9H, 0A4H, 0B0H, 99H, 92H, 82H, 0F8H, 80H, 90H	;Decimal 0 to 9 hex code stored at code memory starting at location 0200H onward
			END	

Problem statement for student: Develop assembly program to display decimal numbers 9 to 0 on Common Anode/ Common Cathode 7-segment display

Step 1-Algorithm	Step 2-Flowchart
-------------------------	-------------------------

Step 3- Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments

X I Resources used

Sr. No.	Name of Resource	Specifications	Quantity

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

1.
2.
3.
4.
5.
6.
7.
8.

XIII Observations for Problem statement (use blank sheet provided if space not sufficient)

Sr. NO.	Memory Location	Hex Value
1	C:0x0300	
2	C:0x0301	
3	C:0x0302	
4	C:0x0303	
5	C:0x0304	
6	C:0x0305	
7	C:0x0306	
8	C:0x0307	
9	C:0x0308	
10	C:0x0309	

XIV Results (Output of the Program)

.....

XV Interpretation of Results (Give meaning of the above obtained results)

.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....

XVII Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifies CO.

1. Write the 7-segment hex code to display letter 'E' and 'F' on Common Anode display.
2. State number of pins a seven segment display IC have.
3. Write assembly language program to display numbers 4 and 5 alternately on common cathode seven segment display

[Space for Answers]

.....

.....

.....

.....

.....

.....

.....

.....

.....

XVIII References/Suggestions for further reading

1. <https://www.geeksforgeeks.org/seven-segment-displays/>
2. <https://www.electronics-tutorials.ws/blog/7-segment-display-tutorial.html>
3. <https://www.circuitstoday.com/interfacing-seven-segment-display-to-8051>

XIX Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 19: Relay interfacing to Microcontroller

I Practical Significance

Electromagnetic or solid-state relays are used in electronic applications to switch or control high voltages or high currents. In Industrial applications low power devices microcontrollers drive relays are used to control electrical loads beyond their direct drive capability. Electromechanical protective relays are used to detect overload and other faults on electrical lines by opening and closing circuit breakers. This practical will help the students to develop skills to interface relay to microcontroller and turn it ON and OFF.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Interface memory and I/O peripherals to 8051 microcontroller.

IV Laboratory Learning Outcome(s)

Interface relay with microcontroller and turn it ON and OFF.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

Turning a relay on and off is a fundamental function for controlling power to a device or circuit using a separate, usually lower voltage circuit. A relay is an electromechanical switch that uses an electromagnet to mechanically operate a switch.

Turning a Relay On:

1. To turn on the relay, a current must be applied to the coil. This can be done using a control signal from a microcontroller, a switch, or any digital output capable of driving the relay.
2. This input activates the coil, causing the contacts to close and allowing power to pass through to the connected device.

Turning a Relay Off:

1. To turn off the relay, the current flowing through the coil is interrupted. This can be achieved by cutting off the control signal or through a switch.
2. When the coil is de-energized, a spring or other mechanism forces the contacts back into the open position, cutting off power to the device.

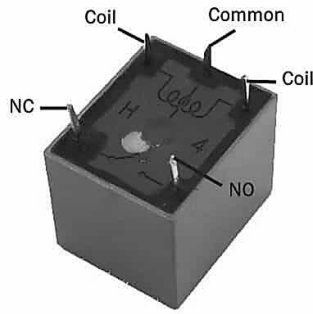


Fig 19.1 Relay Terminals

VII Actual Circuit Diagram used in laboratory

a) Sample Circuit Diagram

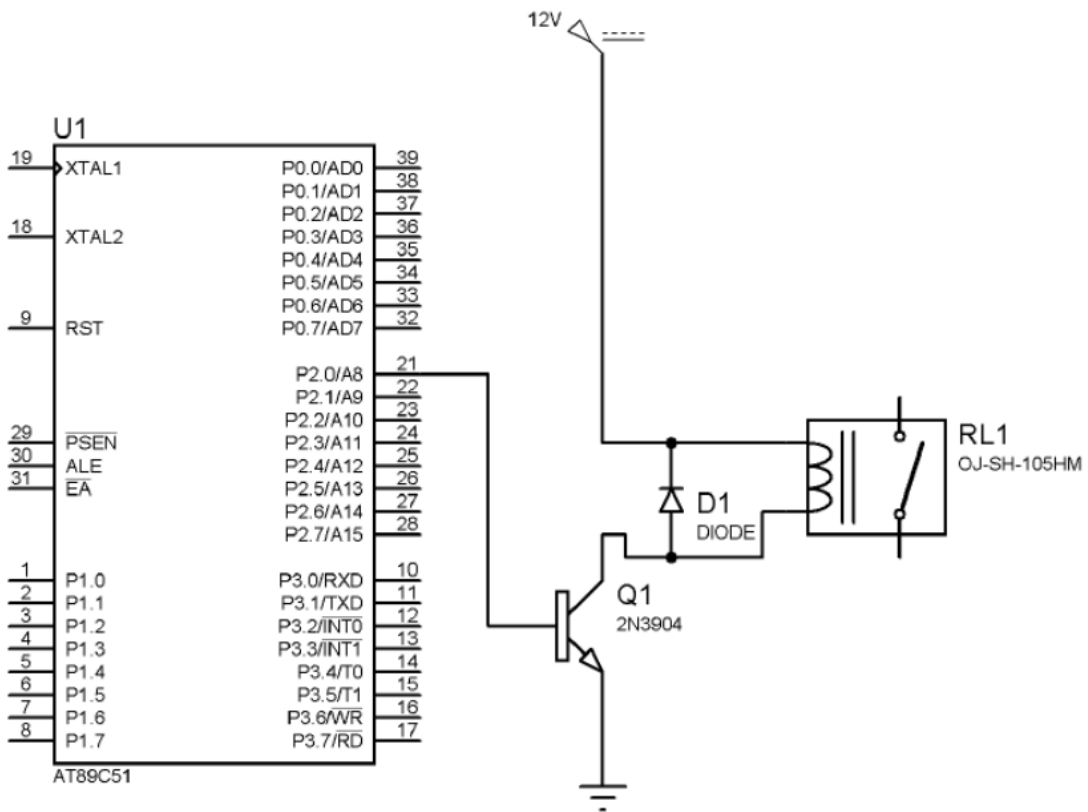


Fig 19.2 Sample Circuit Diagram

b) Practical Setup:

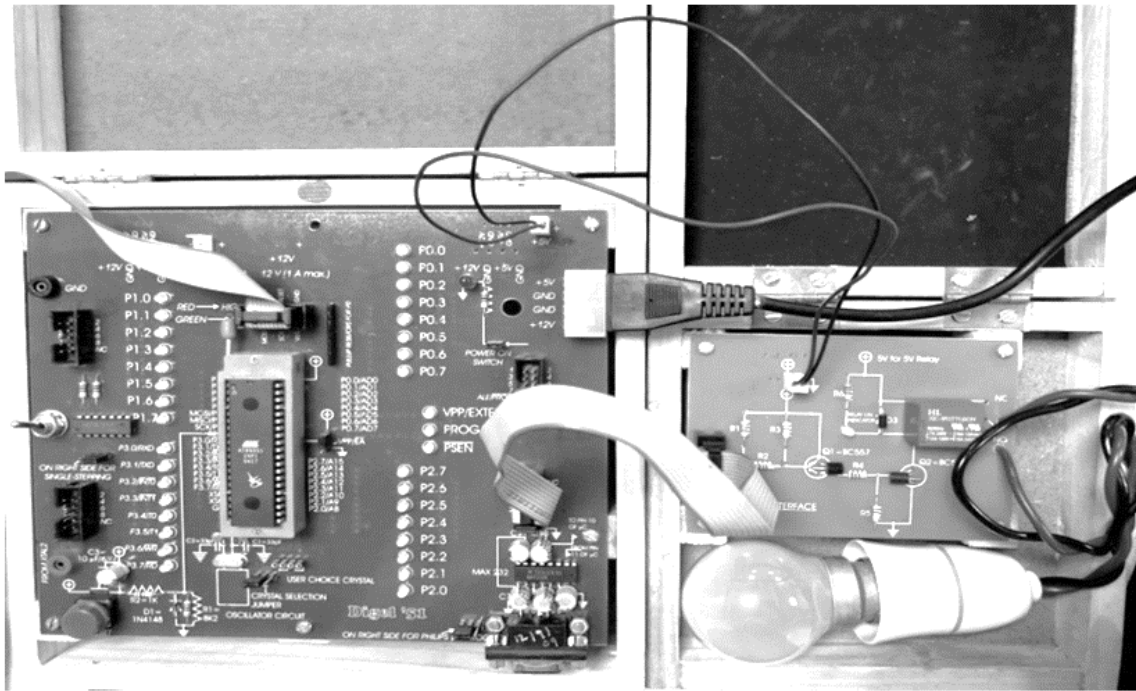


Fig 19.3 Practical Set up Diagram

c) Actual Circuit Diagram used:

VIII Resources Required

Sr. No	Instrument /Components	Specification	Quantity
1.	Microcontroller kit	Single board systems with 8K RAM, ROM memory with battery backup, 16X4, 16X2 LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross c-compiler, RS-232, USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open-source IDE, simulation & program downloading software	
3	Relay trainer board	Suitable to interface with 8051 trainer kit	1 No.

IX Precautions to be Followed

1. Check rules / syntax of assembly programming.
2. Use always driver circuit before interfacing relay to the microcontroller.
3. Use fly back diode to avoid voltage spikes

X Procedure**Write Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write program to display various patterns on 4x4 LED matrix.

Step 1: Algorithm

1. Make the Port pin P1.0 used to Interface relay as an output pin.
2. Turn on relay by setting port bit.
3. Add delay

4. Turn off relay by clearing bit
5. Add delay
6. For repeat operation go to step 2
7. Stop

Step 2: Flowchart

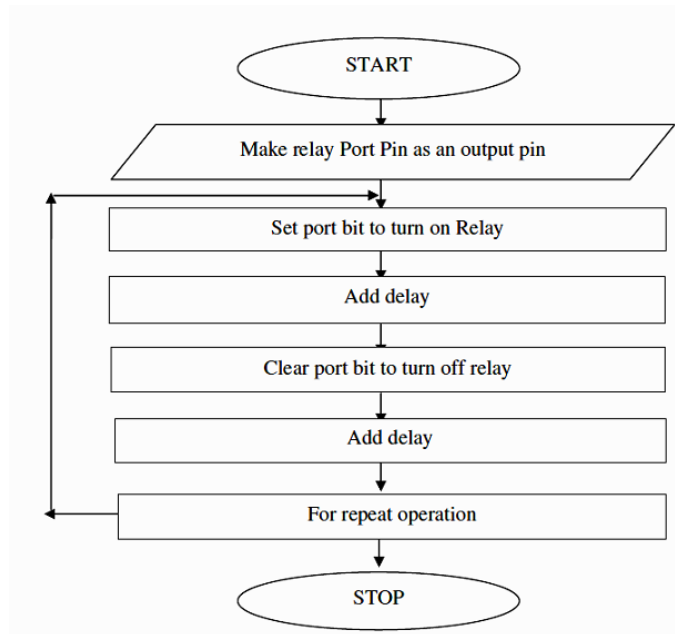


Fig . 19.4 Flowchart for 4x4 LED matrix

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0x0000	D290	MAIN	SETB P2.0	Set P2.0 high to turn ON the relay
C:0x0002	110A		ACALL DELAY	Call Delay
C:0x0004	C290		CLR P2.0	Clear P2.0 low to turn OFF the relay
C:0x0006	110A		ACALL DELAY	Call Delay
C:0x0008	80F6		SJMP MAIN	Infinite loop to toggle relay
C:0x000A	78FF		MOV R0, #255	
C:0x000C	79FF	LOOP 1	MOV R1, #255	
C:0x000E	D9FE	LOOP 2	DJNZ R1, LOOP 2	Decrement and jump if not zero
C:0x0010	D8FA		DJNZ R0, LOOP 1	
C:0x0012	22		RET	
			END	

Problem statement for student: Develop assembly program to turn ON and OFF all Relay connected to port 2 with 30 msec delay.

Step 1-Algorithm	Step 2-Flowchart
-------------------------	-------------------------

Step 3- Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XVIII References / Suggestions for further reading

1. <https://www.etechnog.com/2021/07/relay-wiring-diagram-and-function.html>
2. <https://bravelearn.com/turn-relay-on-or-off-using-8051-microcontroller-at89c51/>
3. <https://electrosome.com/interfacing-relay-8051-keil-c/>

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40% (10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 % (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 20: LCD interfacing to 8051 to display characters and decimal numbers

I Practical Significance

Display units are the most important output devices in embedded projects and electronics products. 16x2 LCD is one of the most used display unit. LCDs find application in consumer appliances such as CD players, DVD players, digital watches, computers, etc. These are commonly used in the screen industries. This practical will help the students to develop skills to interface LCD with microcontroller to display the given integer and character.

II Industry/Employer Expected Outcome(s)

Maintain microcontroller based systems.

III Course Level Learning Outcome(s)

Interface the memory and I/O peripherals to 8051 microcontroller

IV Laboratory Learning Outcome(s)

Interface LCD with 8051 microcontroller to display the characters and decimal numbers.

V Relevant Affective Domain related outcome(s)

1. Follow safe practices.
2. Maintain tools and equipment.
3. Follow ethical practices.

VI Relevant Theoretical Background

A 16x2 LCD display is a very basic module and is very commonly used in various devices and circuits. A 16x2 LCD means it can display 16 characters per line and there are 2 such lines.



Fig 20.1 16 X 2 LCD display

HD44780 16 X2 LCD Features

- The operating voltage is 4.7V-5.3V
- It includes two rows where each row can produce 16-characters.
- The utilization of current is 1mA with no backlight
- Every character can be built with a 5×8 pixel box
- The alphanumeric LCDs display alphabets & numbers
- It can work on two modes like 4-bit & 8-bit
- Available in Blue & Green Backlight

LCD Pin functions:**Table 20.1 LCD pin Functions**

PIN NO	NAME	FUNCTION
1	GND	Ground pin.
2	VCC	Power supply pin of 5V.
3	VEE	Used for adjusting the contrast, commonly attached to the potentiometer.
4	RS	RS is the register select pin If RS = 0, the instruction command code register is selected. If RS = 1, the data register is selected
5	R/W	R/W - read/write: R/W input allows the user to write information to the LCD or read information from it. R/W = 1 when reading; R/W = 0 when writing.
6	E	Enable pin is used by the LCD to latch the information present on the data pins. A high-to-low pulse is needed to latch the data. This pulse must be a minimum of 450 ns wide.
7	D0	D0-D7 Data pins for giving data (normal data like numbers characters or command data) which is meant to be displayed.
8	D1	
9	D2	
10	D3	
11	D4	
12	D5	
13	D6	
14	D7	
15	LED+	Back light Anode which should be connected to Vcc(5V)
16	LED-	Back light Cathode which should be connected to ground.

LCD Commands:

Table 20.2 LCD Commands

Sr.No.	Command Code (Hex)	Instruction
1	0x30	Function Set: 8-bit, 1 Line, 5x7 Dots
2	0x38	Function Set: 8-bit, 2 Line, 5x7 Dots
3	0x20	Function Set: 4-bit, 1 Line, 5x7 Dots
4	0x28	Function Set: 4-bit, 2 Line, 5x7 Dots
5	0x06	Entry Mode
6	0x08	Display off Cursor off
7	0x0E	Display on Cursor on
8	0x0C	Display on Cursor off
9	0x0F	Display on Cursor blinking
10	0x18	Shift entire display left
11	0x1C	Shift entire display right
12	0x10	Move cursor left by one character
13	0x14	Move cursor right by one character
14	0x01	Clear Display (also clear DDRAM content)
15	0x80 + address*	Set DDRAM address or cursor position on display
16	0x40 + address**	Set CGRAM address or set pointer to CGRAM location

Note: For more details of the commands refer Data sheet of LCD.

VII Practical Circuit diagram:

a) Sample Circuit diagram

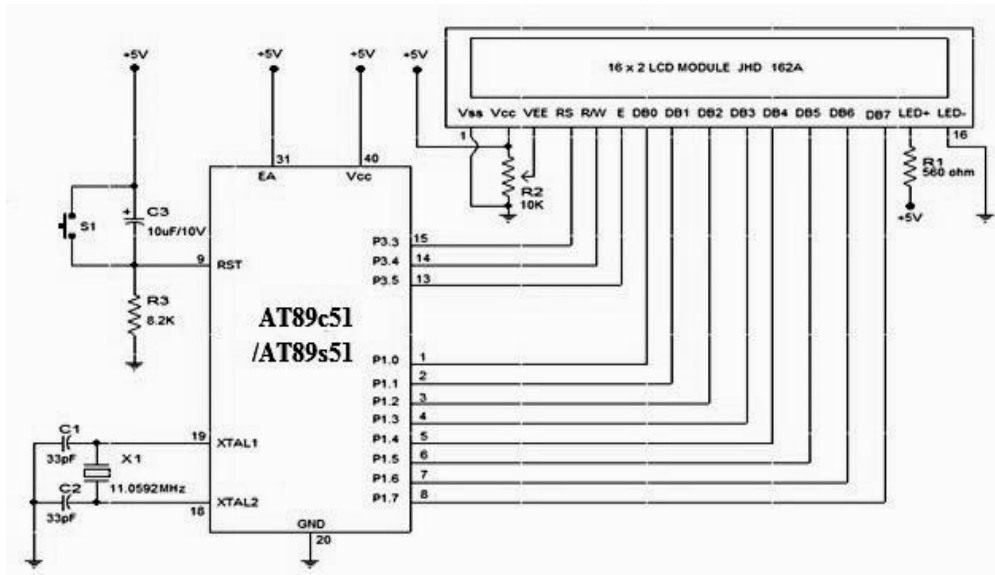


Fig 20.2 8051 connection to 16 X 2 LCD display

b) Practical Setup

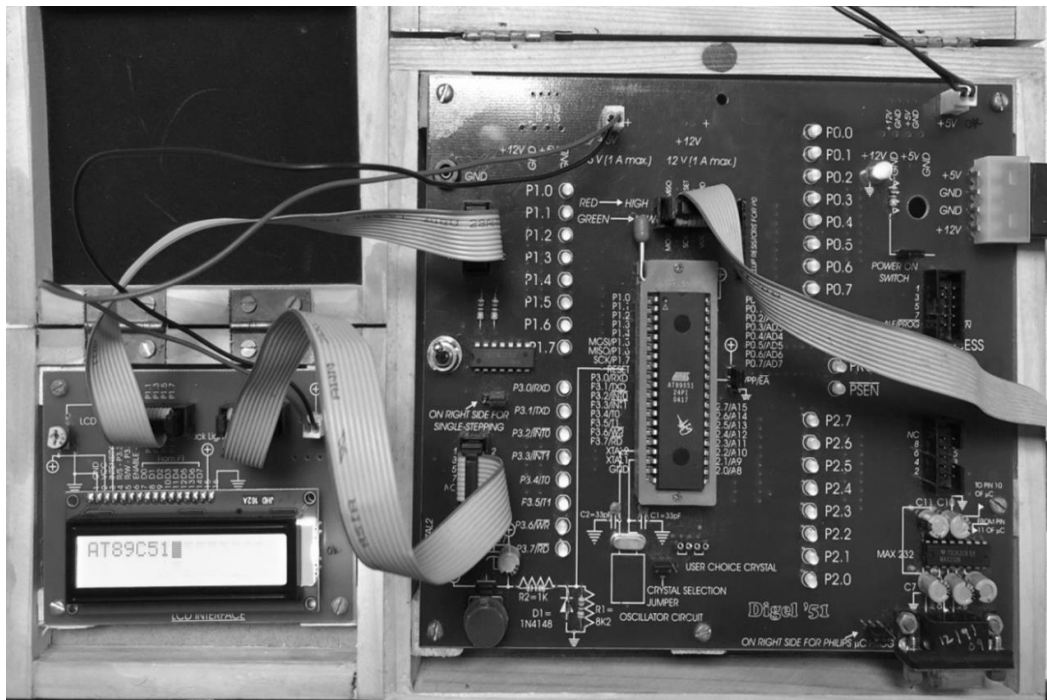


Fig 20.3 Practical setup

c) Simulation diagram

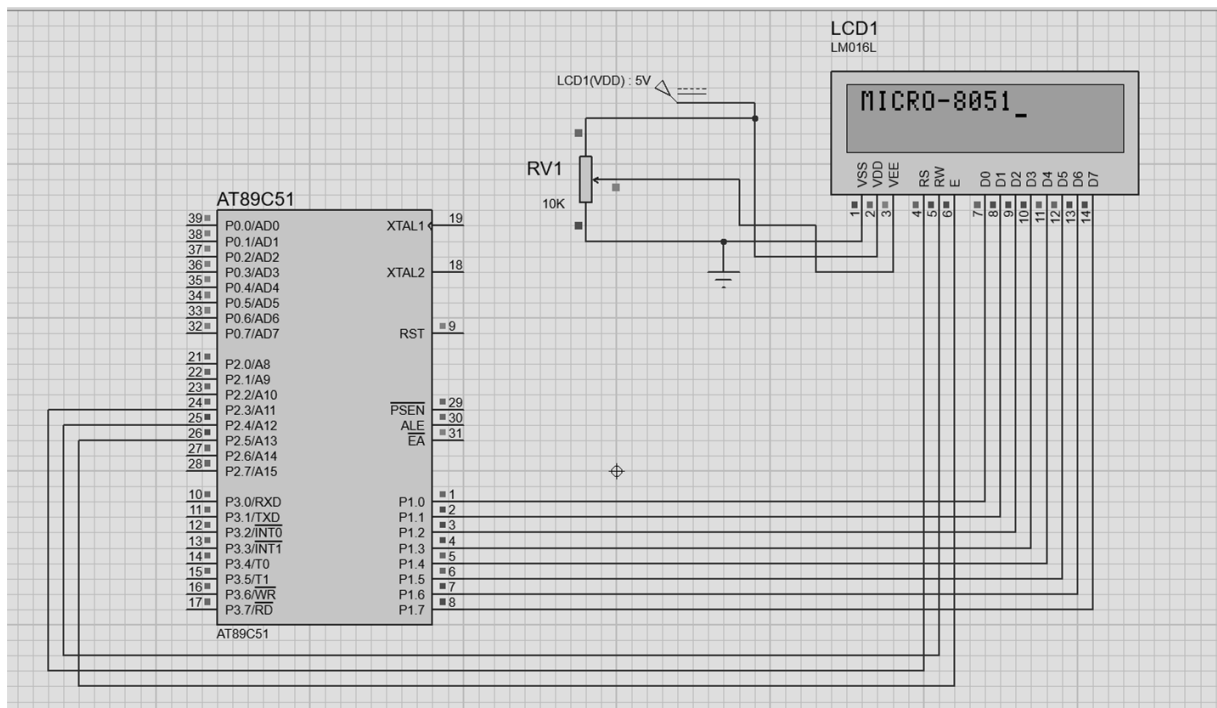


Fig 20.4 Simulation diagram

d) Actual circuit used in Laboratory

e) Actual Experimental set up used in laboratory

VIII Required Resources/apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1.	Microcontroller kit	Single board system with 8K RAM,ROM memory with battery backup,16X4,16X2LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross C-compiler,RS-232,USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open source IDE, simulation and program downloading software.	1 No.
3.	LCD Trainer board	Suitable to interface with 8051 trainer kit	1 No.

IX Precautions to be followed

- 1) Ensure proper connection before turning ON power supply to the kit.
- 2) Don't apply pressure to the display surface.
- 2) Check rules / syntax of assembly language programming.

X Procedure

1. Write algorithm for given problem.
2. Draw flowchart for the same.
3. Develop assembly program using Integrated Development Environment (IDE) or any other relevant software tool.
4. Debug program on IDE.
5. Execute program on IDE.
6. Create hex file for the above program.
7. Interface LCD display to microcontroller as per circuit diagram shown in Fig 20.2
8. Download hex code in EPROM/Flash memory of microcontroller
9. Observe output on LCD display

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM : Write a program to display “MICRO-8051” on LCD.

Step 1-Algorithm

1. Define control signals RS, RW and EN for LCD
2. Make LCD connected port as an output port.
3. Initialize LCD by sending commands.
4. Load DPTR with program memory address.
5. Set register as counter R2 =10(decimal) for to display “MICRO-8051”
6. Clear Accumulator
7. Read ASCII from code memory into Accumulator.
8. Send code to output port where LCD is connected.
9. Increment memory pointer.
10. Decrement R2 counter. Is count $\neq 0$, then go to step 6.
11. Stop

Step 2: Flowchart

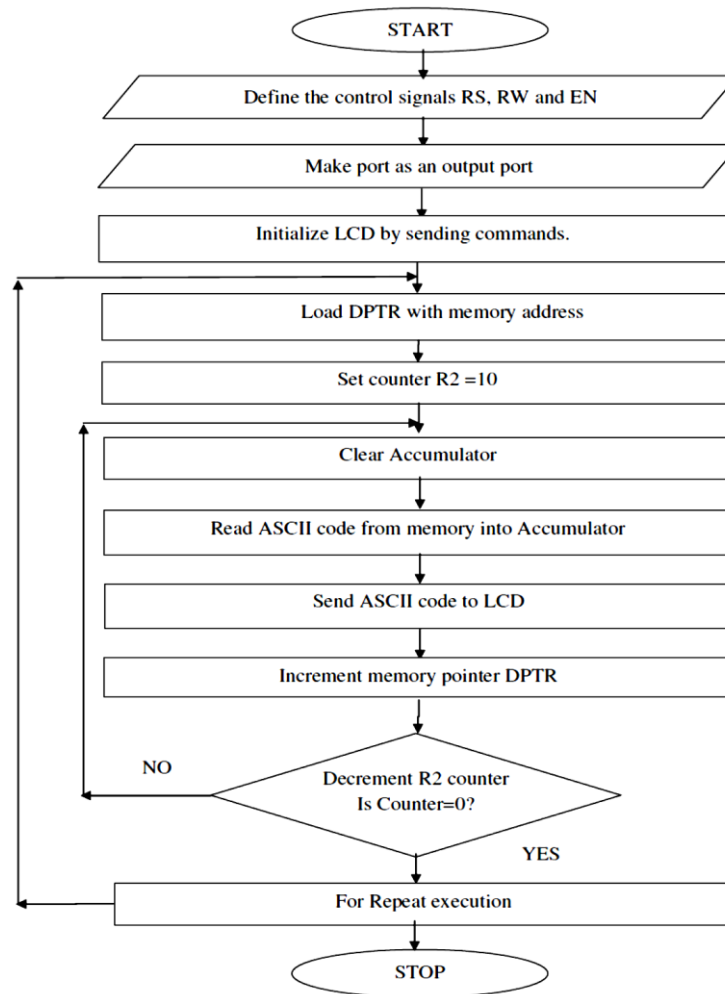


Fig 20.5 Flowchart to display alphabets and decimal numbers on LCD

Step 3: Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			RS EQU P2.3	;Replace a bit address by a symbol
			RW EQU P2.4	
			EN EQU P2.5	
			ORG 0000H	
C:0x0000	759000		MOV P1, #00H	;Set P1 as o/p port where LCD is connected
C:0x0003	120015		LCALL LCD_INIT	;Call LCD initialize subroutine
C:0x0006	900100		MOV DPTR, #MSG	; Load program memory address into DPTR
C:0x0009	7A0A		MOV R2, #10	;Set counter of 10
C:0x000B	E4	UP:	CLR A	
C:0x000C	93		MOVC A,@A+DPTR	;Read data from memory into A register
C:0x000D	120037		LCALL DATAWRT	
C:0x0010	A3		INC DPTR	;Increment pointer to next location
C:0x0011	DAF8		DJNZ R2, UP	;Repeat loop for 10 times
C:0x0013	80FE	HERE:	SJMP HERE	; stop
C:0x0015	7438	LCD_INIT:	MOV A, #38H	; 2 lines and 5×7 matrix (8-bit mode)
C:0x0017	112A		ACALL CMD	
C:0x0019	740E		MOV A, #0EH	; Display on, cursor on
C:0x001B	112A		ACALL CMD	
C:0x001D	7406		MOV A, #06H	; Increment cursor (shift cursor to right)
C:0x001F	112A		ACALL CMD	
C:0x0021	7401		MOV A, #01H	; Clear display screen
C:0x0023	112A		ACALL CMD	
C:0x0025	7480		MOV A, #80H	; Force cursor to beginning to 1st line
C:0x0027	112A		ACALL CMD	
C:0x0029	22		RET	
C:0x002A	F590	CMD:	MOV P1,A	;Send command to lcd
C:0x002C	C2A3		CLR RS	;Select command register
C:0x002E	C2 A4		CLR RW	;Select write operation
C:0x0030	D2A5		SETB EN	
C:0x0032	C2A5		CLR EN	;Latch command to lcd
C:0x0034	1144		ACALL DELAY	;Wait for sometime
C:0x0036	22		RET	

C:0x0037	F590	DATAWRT:	MOV P1,A	;Send data to lcd
C:0x0039	D2 A3		SETB RS	;Select data register
C:0x003B	C2 A4		CLR RW	;Select write operation
C:0x003D	D2A5		SETB EN	
C:0x003F	C2A5		CLR EN	;Latch data to lcd
C:0x0041	1144		ACALL DELAY	;Wait for sometime
C:0x0043	22		RET	
C:0x0044	7B32	DELAY:	MOV R3,#50	;Delay subroutine
C:0x0046	7CFF	L2:	MOV R4,#255	
C:0x0048	DCFE	L1:	DJNZ R4,L1	
C:0x004A	DBFA		DJNZ R3,L2	
C:0x004C	22		RET	
			ORG 0100H	
		MSG:	DB "MICRO-8051"	;Define data byte to code memory
			END	

Problem statement for student: Develop assembly language program to display "MSBTE" on second line of LCD

<p>Step 1-Algorithm</p>	<p>Step 2-Flowchart</p>
--------------------------------	--------------------------------

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

1.
2.
3.
4.
5.
6.
7.
8.
9.

XIII Observations for Problem statement (use blank sheet provided if space not sufficient)

Sr. No.	LCD Memory Location	Observed Character
1		
2		
3		
4		
5		
6		
7		

XIV Results (Output of the Program)

.....
.....

XV Interpretation of Results (Give meaning of the above obtained results)

.....
.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....
.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XVIII References/Suggestions for further reading

1. <https://circuitdigest.com/article/16x2-lcd-display-module-pinout-datasheet>
2. <https://www.dnatechindia.com/Interfacing-LCD-to-8051.html>
3. <https://circuitdigest.com/microcontroller-projects/lcd-interfacing-with-8051-microcontroller-89s52>
4. <https://www.elprocus.com/lcd-interfacing-with-8051-microcontroller/>

XIX Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 21: Keyboard interfacing to 8051

I Practical Significance

Keyboards allow users to input data directly into an embedded system. This interaction is essential for devices that require user configuration, command entry, or data input, enhancing usability and functionality. A keyboard interface with an 8051 microcontroller can be customized to handle various types of keyboards and input methods. This flexibility allows developers to design interfaces that best suit their specific application needs. This practical will help the students to develop skills to interface given keyboard to the microcontroller and display key pressed.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Interface memory and I/O peripherals to 8051 microcontroller.

IV Laboratory Learning Outcome(s)

Interface the given keyboard with 8051 and display the key pressed.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

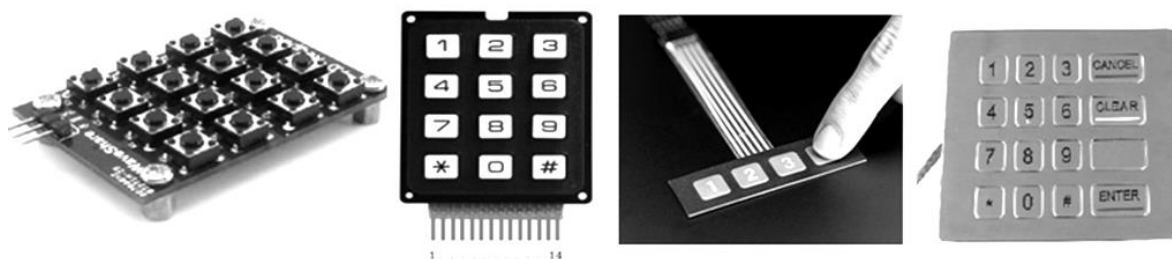


Fig 21.1 Types of Keyboards

4 x 4 matrix keypad connected to a single port of microcontroller. The keypad columns and rows are connected to the port pins. The keypad can be decoded to find out which key was pressed. When a key is pressed on the keypad, a row and column make a contact; otherwise, there is no connection

Specifications: Keypad

1. Maximum voltage across each key: 24V
2. Maximum Current through each key: 30mA
3. Maximum operating temperature: 0°C to + 50°C
4. Easy interface
5. Long life

**Table 21.1 4x4 Keypad: - Rows (R1, R2, R3, R4) Columns (C1, C2, C3, C4)
(Refer Fig.15.2)**

<p>Step1:- Make R1- 0 Checks C1, C2, C3,C4 If C1=0 – ‘0’ is pressed If C2=0 – ‘1’ is pressed If C3=0 – ‘2’ is pressed If C3=0 – ‘3’ is pressed</p>	<p>Step3:- Make R3- 0 Checks C1, C2, C3,C4 If C1=0 – ‘8’ is pressed If C2=0 – ‘9’ is pressed If C3=0 – ‘A’ is pressed If C3=0 – ‘B’ is pressed</p>
<p>Step2:- Make R2- 0 Checks C1, C2, C3,C4 If C1=0 – ‘4’ is pressed If C2=0 – ‘5’ is pressed If C3=0 – ‘6’ is pressed If C3=0 – ‘7’ is pressed</p>	<p>Step4:- Make R4- 0 Checks C1, C2, C3,C4 If C1=0 – ‘C’ is pressed If C2=0 – ‘D’ is pressed If C3=0 – ‘E’ is pressed If C3=0 – ‘F’ is pressed</p>

VII Actual Circuit Diagram used in laboratory

a) Sample Circuit Diagram

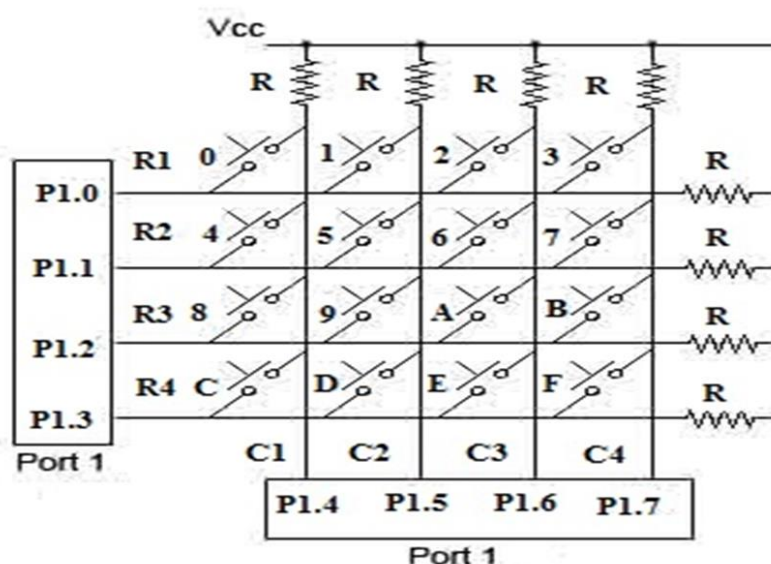


Fig 21.2 Sample Circuit Diagram

b) Practical Setup:

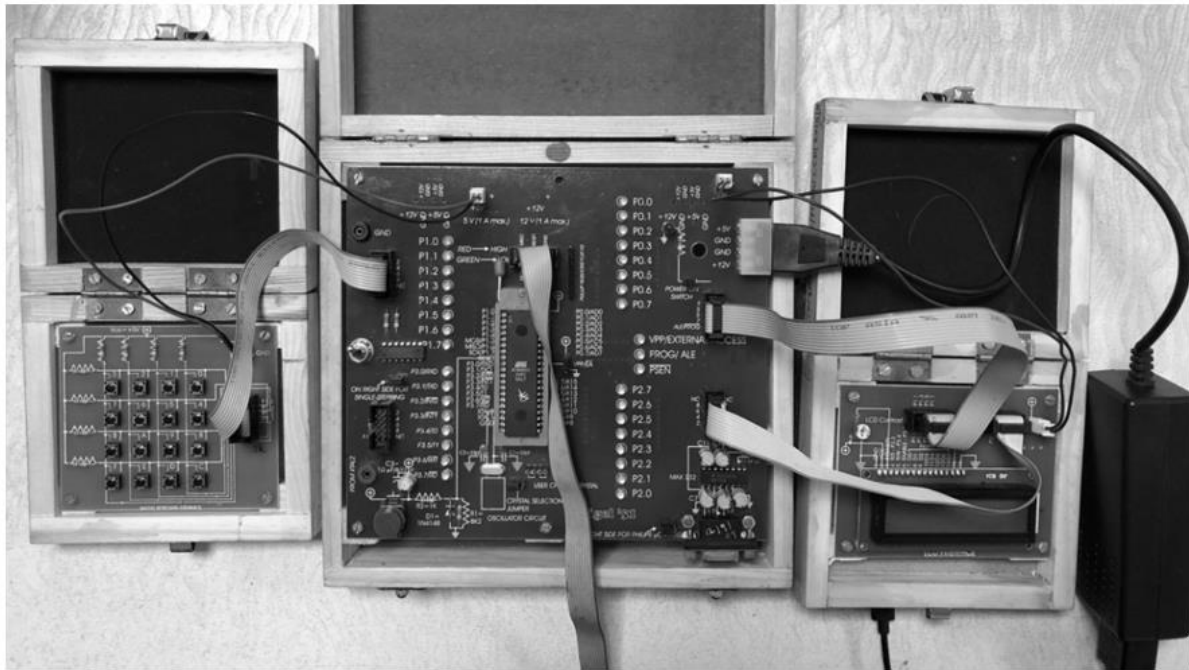


Fig 21.3 Practical Set up Diagram

c) Simulation Diagram:

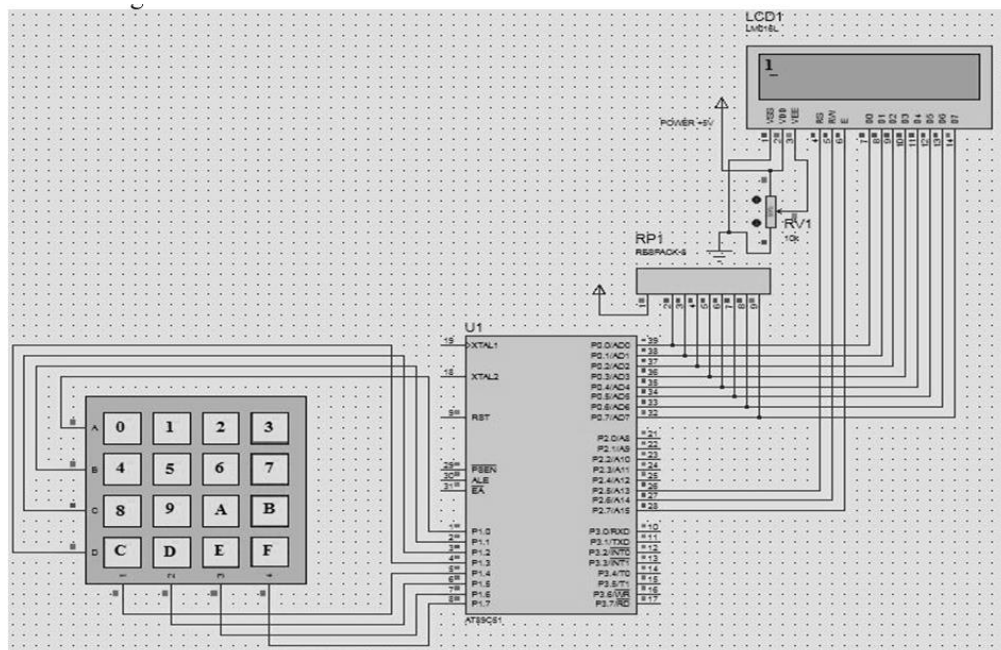


Fig 21.4: Simulation Diagram

d) Actual Circuit Diagram used:**VIII Resources Required**

Sr. No	Instrument /Components	Specification	Quantity
1.	Microcontroller kit	Single board systems with 8K RAM, ROM memory with battery backup,16X4,16X2LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross c-compiler,RS-232,USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open source IDE, simulation and program downloading software	1 No.
3	Keyboard 4x4 trainer board	Suitable to interface with 8051 trainer kit	1No.

IX Precautions to be Followed

1. Check rules / syntax of assembly programming.

X Procedure**Write Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write a program to display key pressed on LCD.

Step1- Algorithm

1. Define LCD control pins RS, RW, EN.
 2. Make LCD Port P0 as an output port.
 3. Make the keypad Port P1 pins as an input port.
 4. Initialize LCD by sending commands.
 5. Make R1 low& read columns C1, C2, C3 and C4
If C1=0 display '0', If C2=0 display '1', If C3=0 display '2', If C4=0 display '3'
 6. Make R2 low& read columns C1, C2, C3 and C4.
If C1=0 display '4', If C2=0 display '5', If C3=0 display '6', If C4=0 display '7'.
 7. Make R3 low& read columns C1, C2, C3 and C4.
If C1=0 display '8', If C2=0 display '9', If C3=0 display 'A', If C4=0 display 'B'
 8. Make R4 low& read columns C1, C2, C3 and C4.
If C1=0 display 'C', If C2=0 display 'D', If C3=0 display 'E', If C4=0 display 'F'
1. Go to step 5 to scan keypad.
 2. Stop

Step 2-Flow Chart

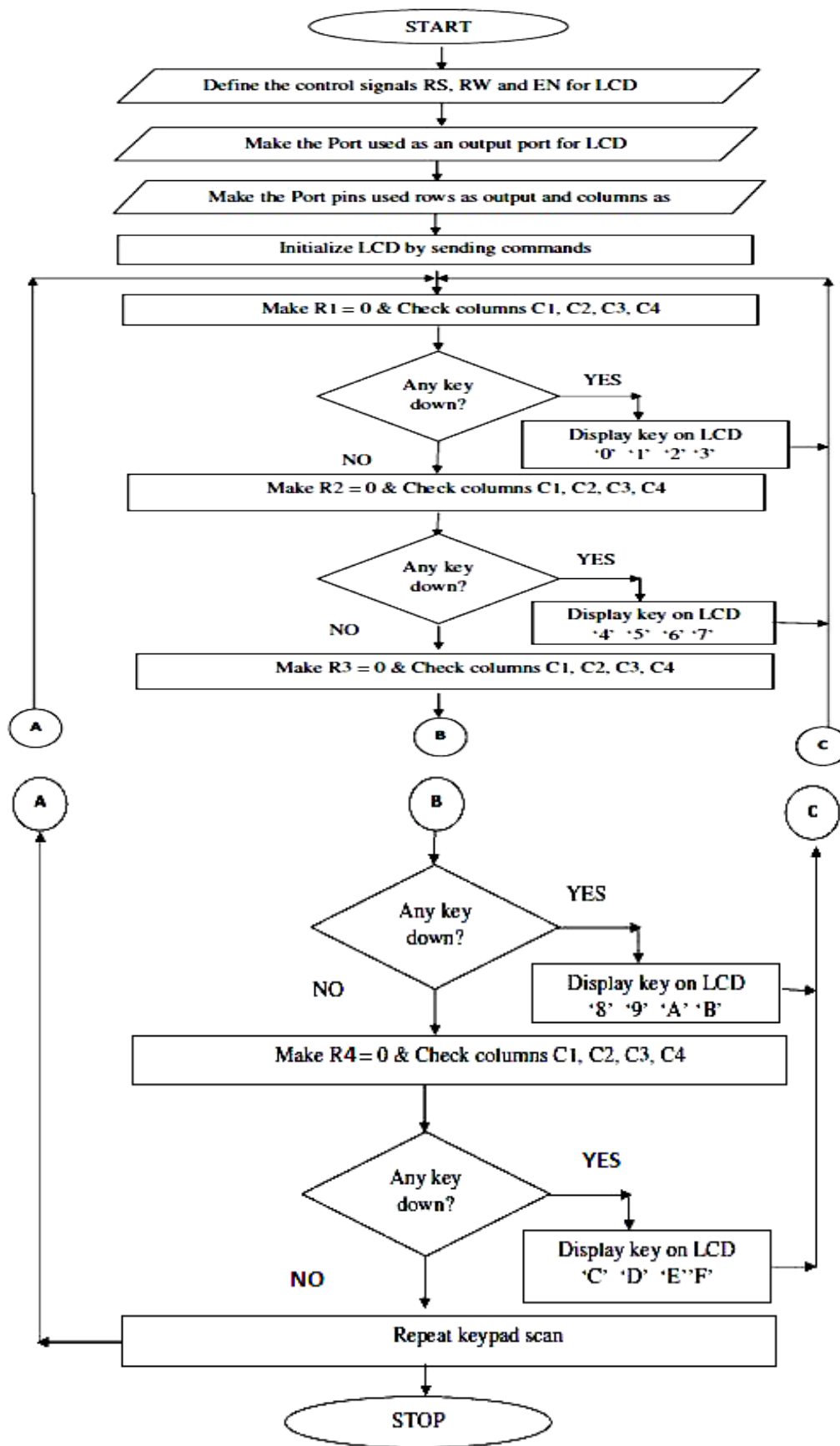


Fig 21.5 Flowchart to display the key pressed on LCD

Step 3-Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ROW1 BIT P1.0	;keypad row pins
			ROW2 BIT P1.1	
			ROW3 BIT P1.2	
			ROW4 BIT P1.3	
			COL1 BIT P1.4	;keypad column pins
			COL2 BIT P1.5	
			COL3 BIT P1.6	
			COL4 BIT P1.7	
			RS BIT P2.5	;LCD control pins
			RW BIT P2.6	
			EN BIT P2.7	
			ORG 0000H	
C:0x0000	758000		MOV P0,#00H	;make LCD port as output
C:0x0003	11AB		ACALL LCDINIT	
C:0x0005	7590FF	KEYSCAN:	MOV P1,#0FFH	;step 1 make port as input
C:0x0008	C290		CLR ROW1	;make row1=0
C:0x000A	30943E		JNB COL1,K0	;check col1
C:0x000D	309441		JNB COL2,K1	;check col2
C:0x0010	309444		JNB COL3,K2	;check col3
C:0x0013	309447		JNB COL4,K3	;check col4
C:0x0016	7590FF		MOV P1,#0FFH	;step 2
C:0x0019	C291		CLR ROW2	;make row2=0
C:0x001B	309445		JNB COL1,K4	
C:0x001E	309548		JNB COL2,K5	
C:0x0021	30964B		JNB COL3,K6	
C:0x0024	30974E		JNB COL4,K7	
C:0x0027	7590FF		MOV P1,#0FFH	;step 3
C:0x002A	C292		CLR ROW3	;make row3=0
C:0x002C	30954C		JNB COL1,K8	
C:0x002F	30954F		JNB COL2,K9	
C:0x0032	309652		JNB COL3,KA	
C:0x0035	309755		JNB COL4,KB	
C:0x0038	7590FF		MOV P1,#0FFH	;step 4
C:0x003B	C293		CLR ROW4	;make row4=0

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x003D	309453		JNB COL1,KC	
C:0x0040	309556		JNB COL2,KD	
C:0x0043	309659		JNB COL3,KE	
C:0x0046	30975C		JNB COL4,KF	
C:0x0049	80BA		SJMP KEYSCAN	;repeat scanning
C:0x004B	7430	K0:	MOV A,#"0"	;key 0 detected
C:0x004D	11CD		ACALL DISPLAY	
C:0x004F	0105		AJMP KEYSCAN	
C:0x0051	7431	K1:	MOV A,#"1"	;key 1 detected
C:0x0053	11CD		ACALL DISPLAY	
C:0x0055	0105		AJMP KEYSCAN	
C:0x0057	7432	K2:	MOV A,#"2"	;key 2 detected
C:0x0059	11CD		ACALL DISPLAY	
C:0x005B	0105		AJMP KEYSCAN	
C:0x005D	7433	K3:	MOV A,#"3"	;key 3 detected
C:0x005F	11CD		ACALL DISPLAY	
C:0x0061	0105		AJMP KEYSCAN	
C:0x0063	7434	K4:	MOV A,#"4"	;key 4 detected
C:0x0065	11CD		ACALL DISPLAY	
C:0x0067	0105		AJMP KEYSCAN	
C:0x0069	7435	K5:	MOV A,#"5"	;key 5 detected
C:0x006B	11CD		ACALL DISPLAY	
C:0x006D	0105		AJMP KEYSCAN	
C:0x006F	7436	K6:	MOV A,#"6"	;key 6 detected
C:0x0071	11CD		ACALL DISPLAY	
C:0x0073	0105		AJMP KEYSCAN	
C:0x0075	7437	K7:	MOV A,#"7"	;key 7 detected
C:0x0077	11CD		ACALL DISPLAY	
C:0x0079	0105		AJMP KEYSCAN	
C:0x007B	7438	K8:	MOV A,#"8"	;key 8 detected
C:0x007D	11CD		ACALL DISPLAY	
C:0x007F	0105		AJMP KEYSCAN	
C:0x0081	7439	K9:	MOV A,#"9"	;key 9 detected
C:0x0083	11CD		ACALL DISPLAY	
C:0x0085	0105		AJMP KEYSCAN	

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x0087	7441	KA:	MOV A,#"A"	;key A detected
C:0x0089	11CD		ACALL DISPLAY	
C:0x008B	0105		AJMP KEYSKAN	
C:0x008D	7442	KB:	MOV A,#"B"	;key B detected
C:0x008F	11CD		ACALL DISPLAY	
C:0x0091	0105		AJMP KEYSKAN	
C:0x0093	7443	KC:	MOV A,#"C"	;key C detected
C:0x0095	11CD		ACALL DISPLAY	
C:0x0097	0105		AJMP KEYSKAN	
C:0x0099	7444	KD:	MOV A,#"D"	;key D detected
C:0x009B	11CD		ACALL DISPLAY	
C:0x009D	0105		AJMP KEYSKAN	
C:0x009F	7445	KE:	MOV A,#"E"	;key E detected
C:0x00A1	11CD		ACALL DISPLAY	
C:0x00A3	0105		AJMP KEYSKAN	
C:0x00A5	7446	KF:	MOV A,#"F"	;key F detected
C:0x00A7	11CD		ACALL DISPLAY	
C:0x00A9	0105		AJMP KEYSKAN	
C:0x00AB	7438	LCDINIT:	MOV A,#38H	;init LCD 2 lines, 5x7 matrix
C:0x00AD	11C0		ACALL COMMAND	
C:0x00AF	740E		MOV A,#0EH	;LCD on cursor on
C:0x00B1	11C0		ACALL COMMAND	
C:0x00B3	7406		MOV A,#06H	;clear LCD command
C:0x00B5	11C0		ACALL COMMAND	
C:0x00B7	7401	CLEAR:	MOV A,#01H	;shift cursor right
C:0x00B9	11C0		ACALL COMMAND	
C:0x00BB	7480		MOV A,#80H	;Force cursor to beginning of 1 st line
C:0x00BD	11C0		ACALL COMMAND	
C:0x00BF	22		RET	
C:0x00C0	F580	COMMAND :	MOV P0,A	;issue command code
C:0x00C2	C2A5		CLR RS	
C:0x00C4	C2A6		CLR RW	
C:0x00C6	D2A7		SETB EN	

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x00C8	11DE		ACALL DELAY	
C:0x00CA	C2A7		CLR EN	
C:0x00CC	22		RET	
C:0x00CD	F580	DISPLAY:	MOV P0,A	;issue data
C:0x00CF	D2A5		SETB RS	
C:0x00D1	C2A6		CLR RW	
C:0x00C3	D2A7		SETB EN	
C:0x00D5	11DE		ACALL DELAY	
C:0x00D7	C2A7		CLR EN	
C:0x00D9	11DE		ACALL DELAY	;add delay
C:0x00DB	11B7		ACALL CLEAR	;LCD clear
C:0x00DD	22		RET	
C:0x00DE	7B32		MOV R3,#50	
C:0x00E0	7CFF		MOV R4,#255	
C:0x00E2	DCFE	DELAY:	DJNZ R4,LOOP1	
C:0x00E4	DBFA	LOOP2:	DJNZ R3,LOOP2	
C:0x00E6	22	LOOP1:	RET	
			END	

XI Resources Used:

S. No.	Instrument /Components	Specification	Quantity
1.			
2.			
3.			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XIII Observations for sample program (use blank sheet provided if space not sufficient)

Sr. No.	Key Pressed	Output on LCD
1	4	
2	5	
3	A	
4	C	
5	F	

XIV Results (Output of the Program)

.....

.....

.....

.....

XV Interpretation of Results (Give meaning of the above obtained results)

.....

.....

.....

.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....

.....

.....

.....

.....

XVIII References / Suggestions for further reading

1. <https://gmostofabd.github.io/8051-Keypad/>
2. <https://www.circuitstoday.com/interfacing-hex-keypad-to-8051>
3. https://www.brainkart.com/article/Interfacing-and-Program-for-Keyboard-to-8051-Microcontroller_7838/

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40% (10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 % (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 22: ADC interfacing to 8051

I Practical Significance

ADC (Analog to digital converter) forms a very essential part in many embedded projects. In industry and automated instruments, the signals sensed and processed by humans are analog signals. Analog-to-digital conversion is the means by which analog signals are converted into digital data that can be processed by computers for various purposes. This practical will help the students to develop skills to interface given ADC to the microcontroller and verify the input and output.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Interface memory and I/O peripherals to 8051 microcontroller.

IV Laboratory Learning Outcome(s)

Interface ADC with 8051 microcontroller and verify input and output.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

An analog to digital converter or ADC, converts an analog signal to a digital signal. An analog signal has a continuously changing amplitude with respect to time. A digital signal, is a stream of 0s and 1s. An ADC maps analog signals to their binary equivalents. To do this, ADCs use various methods like:

1. Flash conversion,
2. Slope integration, or
3. Successive approximation.

Types of ADC:

1. **Serial ADC:** Serial ADC's consisting of just one output pin that delivers the output code one bit at a time.
2. **Parallel ADC:** Parallel ADC's consisting of several output pins that deliver all the bits of the output code at the same time.

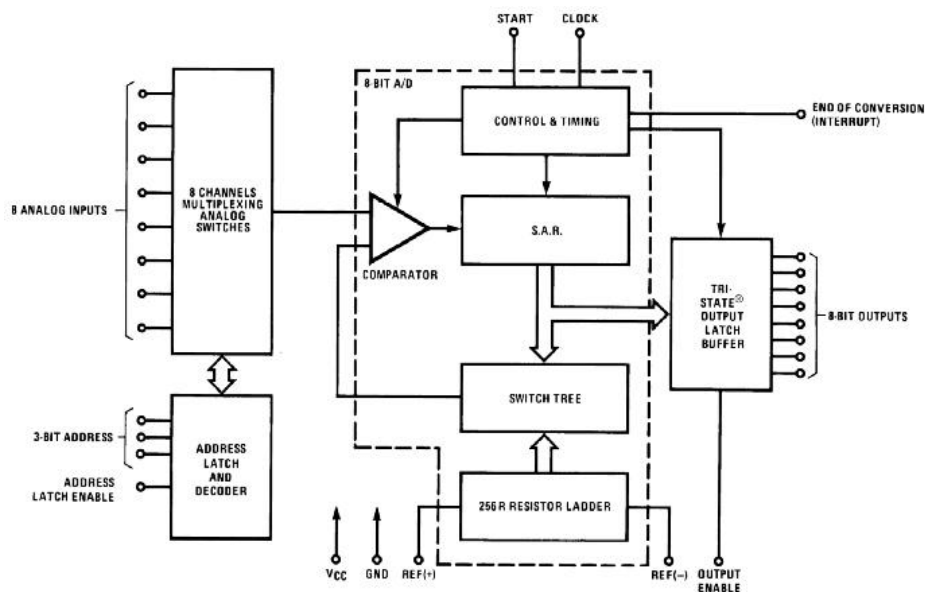


Fig 22.1: ADC Block Diagram

ADC0808 Analog signal selection:

Table 22.1 ADC Channel Selection

Select Lines			Analog Channel Selected
C	B	A	
0	0	0	IN0
0	0	1	IN1
0	1	0	IN2
0	1	1	IN3
1	0	0	IN4
1	0	1	IN5
1	1	0	IN6
1	1	1	IN7

Specifications ADC0808 Chip

- a. ADC0808 IC - analog to digital
- b. Resolution -8 Bits
- c. Input Channels- 8
- d. Single Supply- 5VDC
- e. Low Power- 15mW
- f. Conversion Time -100µs
- g. Output's meets TTL voltage level.
- h. The clock frequency range of ADC is 10 KHz to 1280 KHz.
- i. Typically 680 kHz used.
- j. Low power consumption

VII Actual Circuit Diagram used in laboratory

a) Sample Circuit Diagram

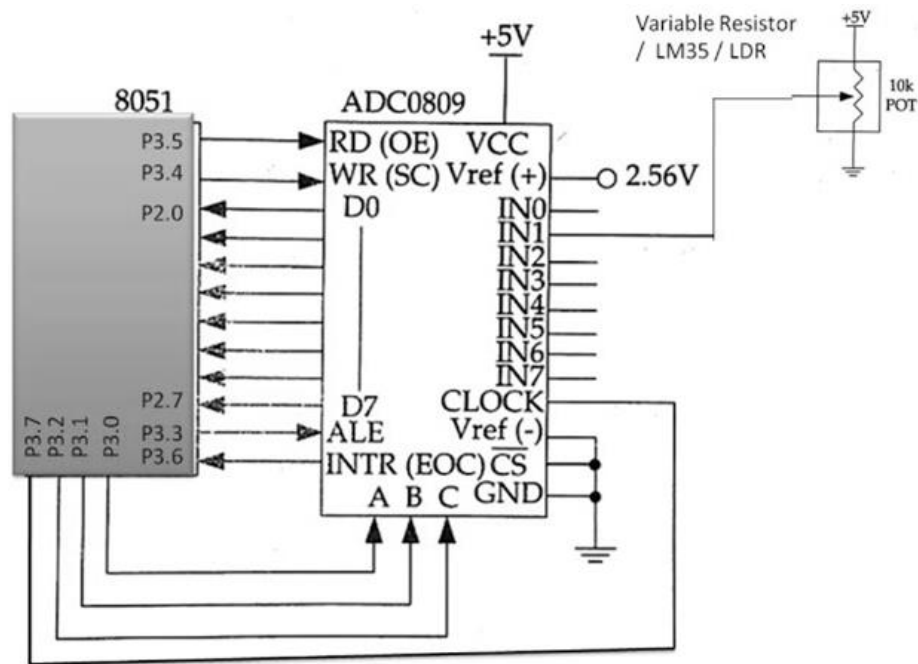


Fig 22.2 ADC interfacing to 8051

b) Practical Setup:

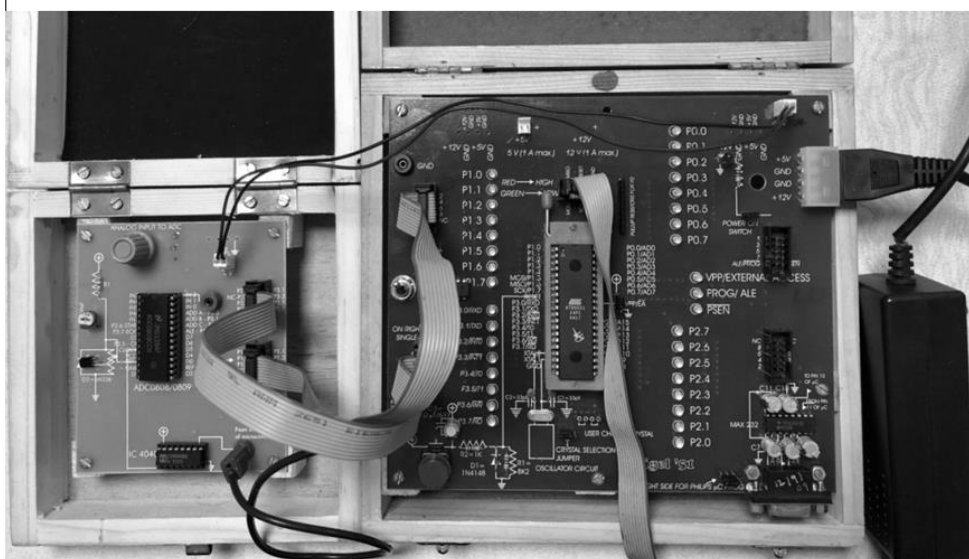


Fig 22.3 Practical Set up Diagram

c) Simulation Diagram:

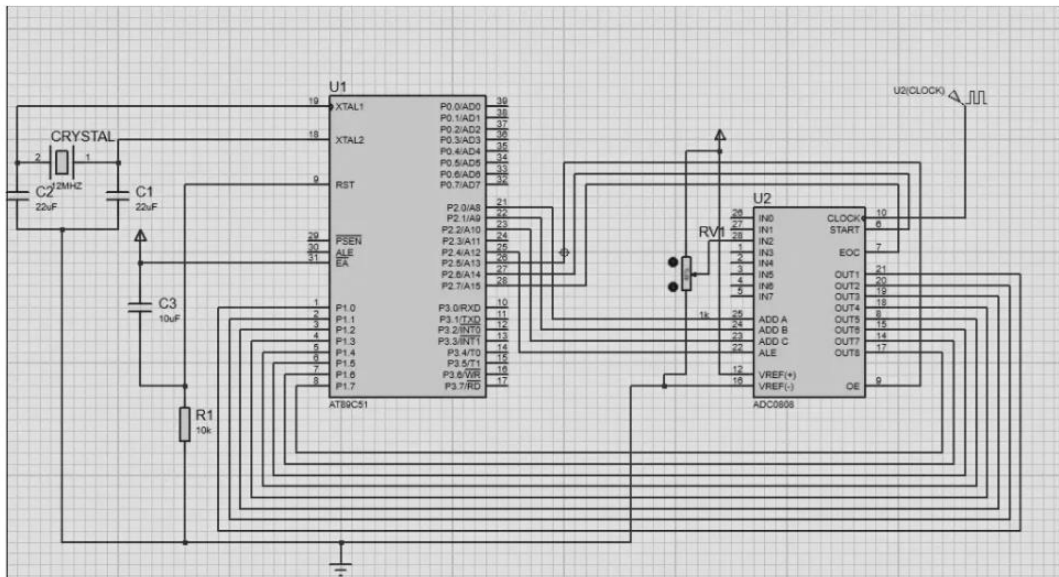


Fig 22.4: Simulation Diagram

d) Actual Circuit Diagram used:

VIII Resources Required

Sr. No	Instrument /Components	Specification	Quantity
1.	Microcontroller kit	Single board systems with 8K RAM, ROM memory with battery backup, 16X4, 16X2 LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross c-compiler, RS-232, USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open-source IDE, simulation and program downloading software	1 No.
3	ADC (0808) trainer board	Suitable to interface 8051 board.	1 No.

IX Precautions to be Followed

1. Check rules / syntax of assembly programming.
2. Refer datasheet for to provide clock frequency to ADC 0808 chip.
3. Care must be taken while taking observations during power up.
4. Use current limiting resistors for LED's.

X Procedure**Write Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write a program to read the data from ADC and display on LEDs.

Step1- Algorithm

1. Select an analog channel by providing bits to A, B, and C addresses according to the analog signal selection table.

2. Activate the ALE (address latch enable) pin.
3. Activate SC (start conversion) to initiate conversion.
4. Monitor EOC (end of conversion) to see whether conversion is finished. H-to L output indicates that the data is converted and is ready to be picked up. If we do not use EOC, we can read the converted digital data after a brief time delay. The delay size depends on the speed of the external clock we connect to the CLK pin.
5. Activate OE (output enable) to read data out of the ADC chip.

Note: In ADC0808 that there is no self-clocking and the clock must be provided from an external source to the CLK pin. Although the speed of conversion depends on the frequency of the clock connected to the CLK pin, it cannot be faster than 100 microseconds.

Step 2-Flow Chart

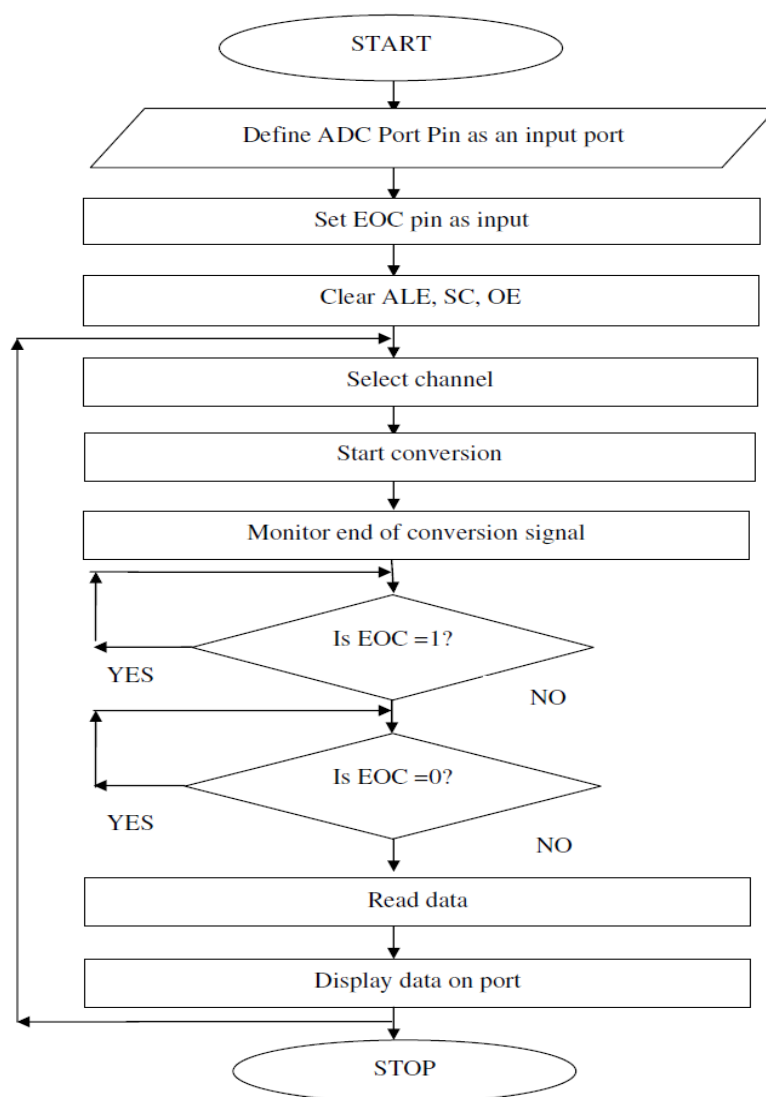


Fig 22.5 Flowchart to display the key pressed on LCD

Step 3-Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	Starting address
C:0X0000	7590FF		MOV P1, #0FFH	Makes port 1 input port
C:0X0003	D2A7		SETB P2.7	Makes EOC pin high
C:0X0005	C2A4		CLR P2.4	Clears ALE pin
C:0X0007	C2A6		CLR P2.6	Clears start pin
C:0X0009	C2A5		CLR P2.5	Clear OE pin
C:0X000B	C2A2	BACK:	CLR P2.2	Clears ADD C
C:0X000D	D2A1		SETB P2.1	Sets ADD B
C:0X000F	C2A0		CLR P2.0	Clears ADD A (this selects the second address line)
C:0X0011	112F		ACALL DELAY	
C:0X0013	D2A4		SETB P2.4	Sets ALE high
C:0X0015	112F		ACALL DELAY	
C:0X0017	D2A6		SETB P.26	Sends a command to start of conversion pulse.
C:0X0019	112F		ACALL DELAY	
C:0X001B	C2A4		CLR P2.4	Makes ALE low
C:0X001D	C2A6		CLR P2.6	Makes start pin low
C:0X001F	20A7FD	HERE:	JB P2.7, HERE	Waits for low pulse at EOC
C:0X0022	30A7FD	HERE1:	JNB P2.7, HERE1	Waits for low pulse to finish
C:0X0025	D2A5		SETB P2.5	Enables OE pin to extract data from ADC
C:0X0027	112F		ACALL DELAY	
C:0X0029	E590		MOV A, P1	Moves acquired data to accumulator
C:0X002B	C2A5		CLR P2.5	Clears OE
C:0X002D	80DC		SJMP BACK	Repeatedly gets data from ADC
C:0X002F	7B32	DELAY:	MOV R3, #50	
C:0X0031	7CFF	HERE2:	MOV R4, #255	
C:0X0033	DCFE		DJNZ R4, HERE3	

XIV Results (Output of the Program)

.....
.....
.....
.....

XV Interpretation of Results (Give meaning of the above obtained results)

.....
.....
.....
.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....
.....
.....
.....

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. List two applications where serial ADC and parallel ADC chips are used.
2. Draw timing diagram for entire ADC process.
3. Distinguish ADC 0804, ADC 0808 and ADC 0848.
4. Give the pin functions of ADC 0808.

[Space for Answers]

.....
.....
.....
.....
.....
.....

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40% (10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 % (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 23: DAC interfacing to generate the square waveform.

I Practical Significance

The digital to analog converter (DAC) is a device widely used to convert digital pulses to analog signals. This practical will help the students to develop skills to interface DAC with 8051 and generate different analog waveforms.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Maintain microcontroller based applications.

IV Laboratory Learning Outcome(s)

Interface DAC with 8051 microcontroller and generate square waveform.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

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.

1. Binary weighted method and
2. R/2R ladder method.

R/2R ladder is the widely used method. This method can achieve a much higher 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. Number of data inputs decides the resolution of DAC. So, if there are n digital input pin, there are 2^n analog levels. So, 8 input DAC has 256 discrete voltage levels.

The output current is known as I_{out} by connecting a resistor to the output to convert into voltage. The total current provided by the I_{out} pin is basically a function of the binary numbers at the input pins D0 - D7 (D0 is the LSB and D7 is the MSB) of DAC0808 and the reference current I_{ref} . The following formula is showing the function of I_{out} :

$$I_{out} = I_{ref} \left(\frac{D_7}{2} + \frac{D_6}{4} + \frac{D_5}{8} + \frac{D_4}{16} + \frac{D_3}{32} + \frac{D_2}{64} + \frac{D_1}{128} + \frac{D_0}{256} \right)$$

The I_{ref} is the input current. This must be provided into the pin 14. Generally, 2.0mA is used as I_{ref} . I_{out} pin is connected to the resistor to convert the current to voltage.

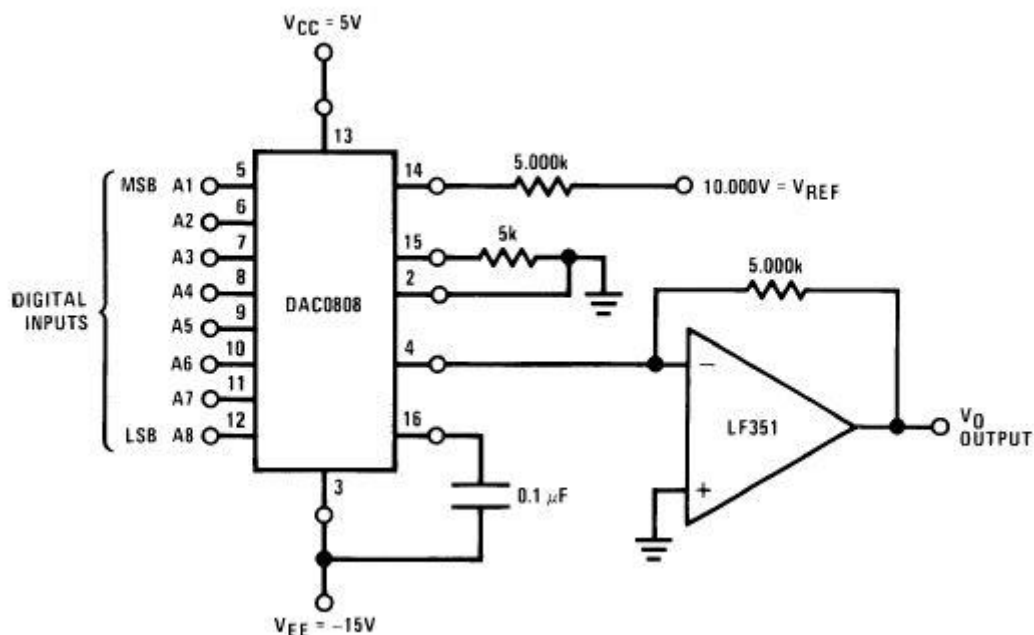


Fig 23.1: DAC Block Diagram

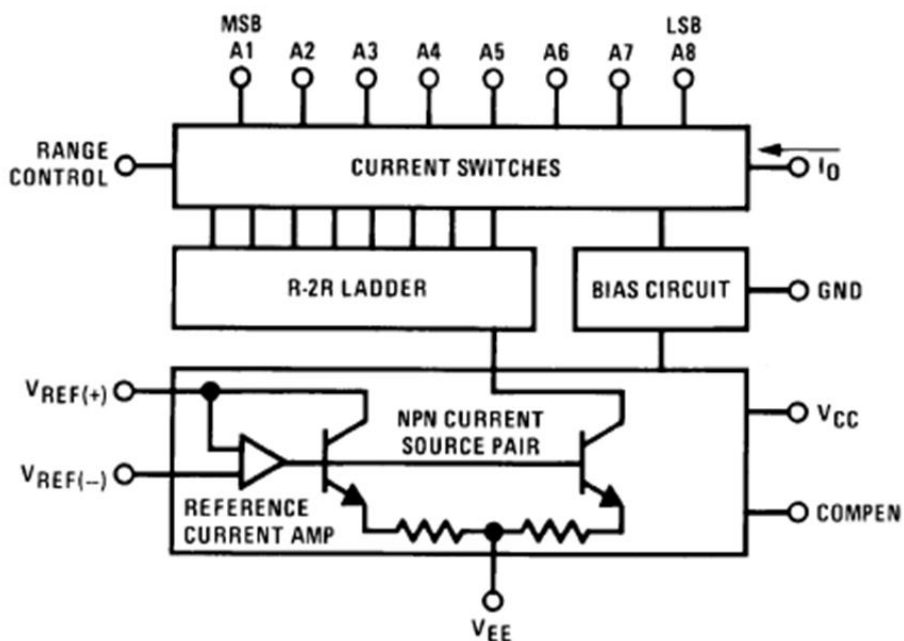


Fig 23.2: DAC Block Diagram

Specifications DAC 0808 Chip

1. Resolution: 8 bits
2. Settling Time: Typically, 100 nanoseconds
3. Output Current: Up to 1.999 mA
4. Voltage Supply: +5V or ±5V to ±18V
5. Accuracy: ±0.19% of full scale
6. Output Type: Voltage output
7. Compatibility: TTL/CMOS

VII Actual Circuit Diagram used in laboratory

a) Sample Circuit Diagram

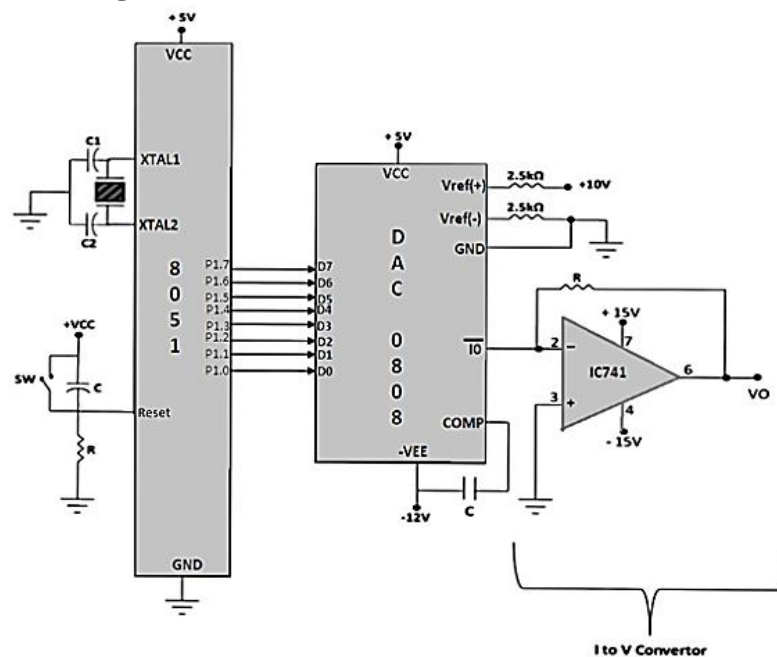


Fig 23.3 DAC interfacing to 8051

b) Practical Setup:

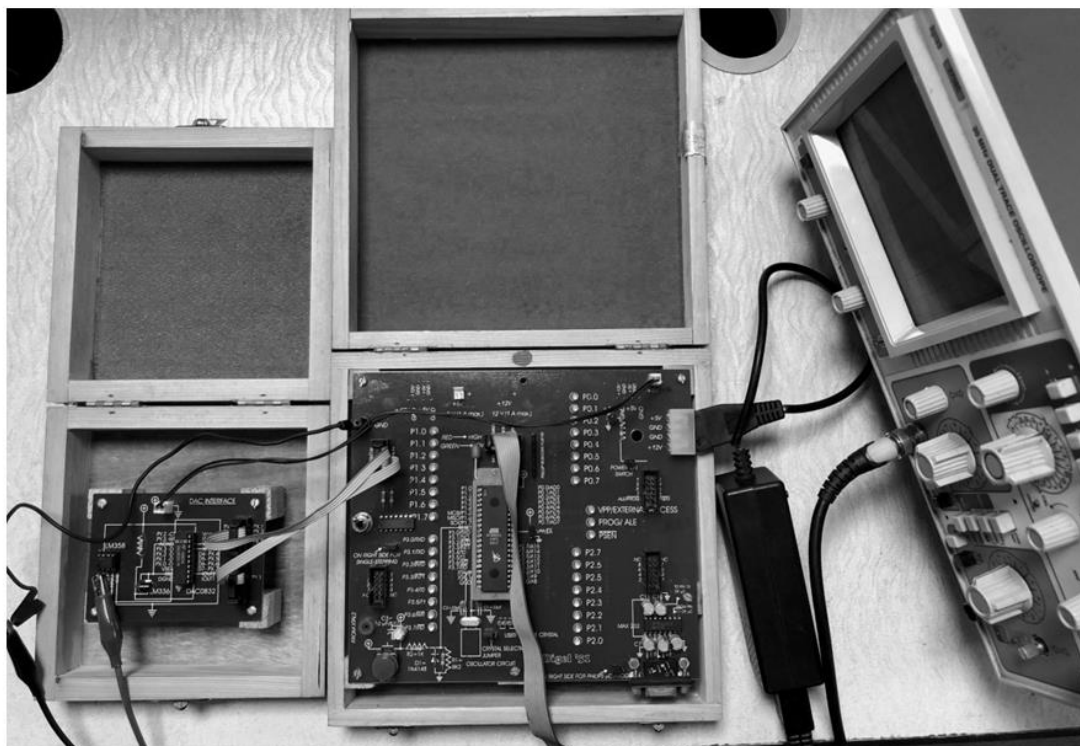


Fig 23.4 Practical Set up Diagram

c) Simulation Diagram:

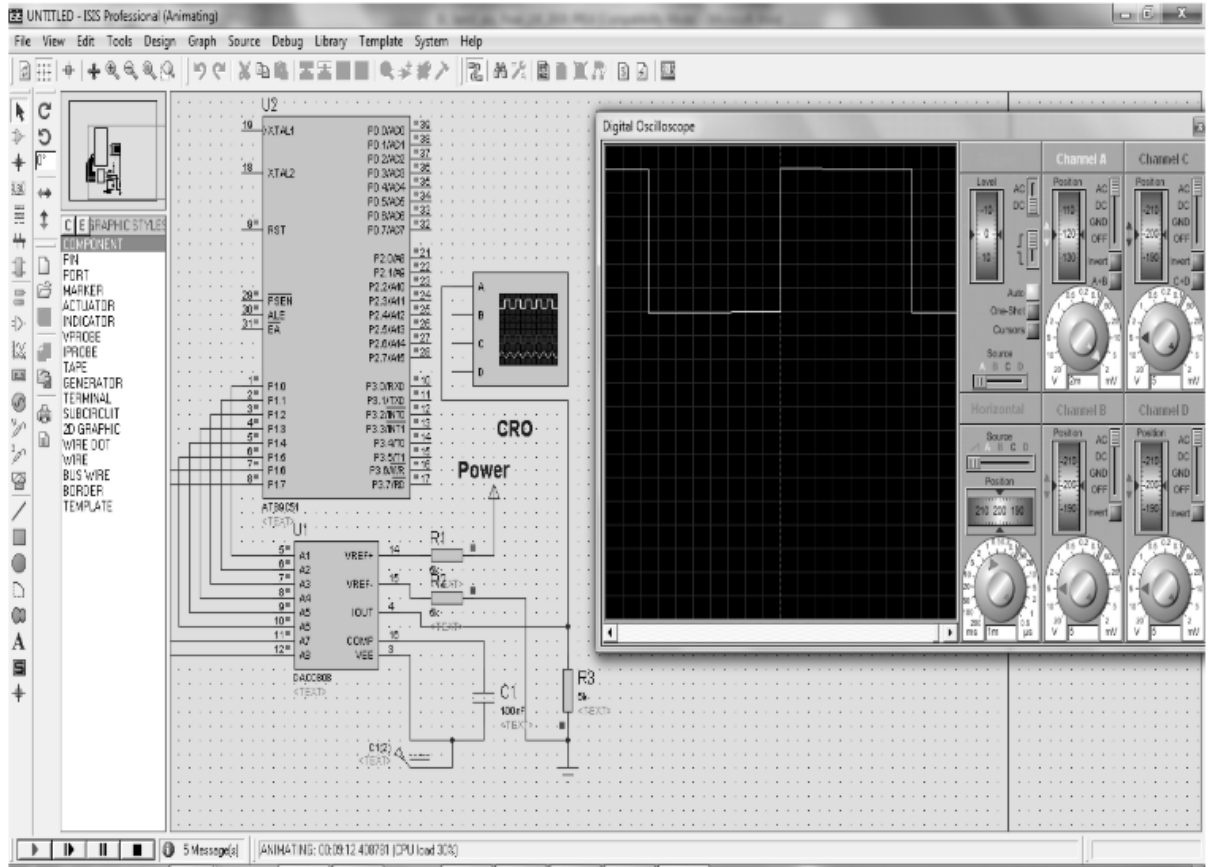


Fig 23.5: Simulation Diagram

d) Actual Circuit Diagram used:

VIII Resources Required

Sr. No.	Instrument /Components	Specification	Quantity
1	Microcontroller kit	Single board systems with 8K RAM,ROM memory with battery backup,16X4,16X2LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross c-compiler,RS-232,USB, interfacing facility with built in power supply.	1 No.
2	Desktop PC	Loaded with open source IDE, simulation and program downloading software	1 No.
3	DAC (0808) trainer board	Suitable to interface 8051 board.	1 No

IX Precautions to be Followed

1. Check rules / syntax of assembly programming.
2. Operate DAC chip as per specifications given in the datasheet otherwise damage may occur to the device.

X Procedure

Write Program

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write a program to generate square waveform using DAC.

STEP1- Algorithm

1. Make the Port used to Interface DAC as an output port.
2. Clear Accumulator.
3. Send 00H value to Port 1

4. Call Delay
5. Send FFH value to port 1
6. Call Delay.
7. For repeat operation go to step3.

Step 2-Flow Chart

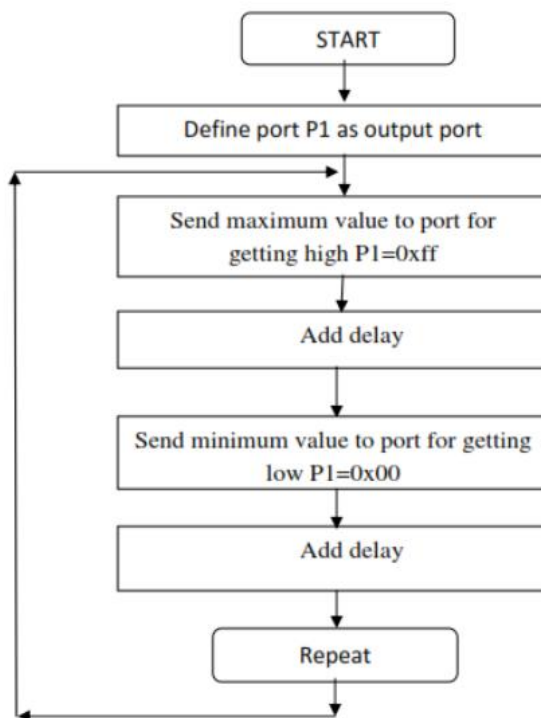


Fig 23.6 Flowchart to generate Square waveform

Step 3-Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000h	
C:0x0000	7400	MAIN:	MOV A, #00h	Clear A
C:0x0002	F590		MOV P1, A	Send value to P1
C:0x0004	110E		ACALL DELAY	Call Delay
C:0x0006	74FF		MOV A, #FFH	Move Highest value in Accumulator
C:0x0008	F590		MOV P1, A	Send value to P1
C:0x000A	110E		ACALL DELAY	Decrement value
C:0x000C			SJMP MAIN	Compare with lowest value
		DELAY		

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x000E	78FF		MOV R0, #100	Initialize Delay register
C:0X0010	79FF	HERE 1:	MOV R1, #100	
C:0x0012	D9FE	HERE:	DJNZ R1, HERE	Repeat till R1 becomes zero
C:0X0014	D8FA		DJNZ R0, HERE1	Repeat till R1 becomes zero
C:0X0016	22		RET	
			END	

Problem statement for student: Develop assembly program to generate a square wave with ON time of 3msec and OFF time of 5 msec. {Assume suitable crystal frequency}

Step 1-Algorithm	Step 2-Flowchart

.....
.....
XV Interpretation of Results (Give meaning of the above obtained results)

.....
.....
.....
.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....
.....
.....
.....

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. If $I_{ref} = 2\text{mA}$ and all the inputs to the DAC are high then find maximum current of DAC 0808 IC.
2. Define Duty cycle for the square waveform.
3. Write a program for generating a square waveform for 60% duty cycle.

[Space for Answers]

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Practical No. 24: DAC interfacing to generate the triangular waveform.

I Practical Significance

The digital to analog converter (DAC) is a device widely used to convert digital pulses to analog signals. This practical will help the students to develop skills to interface DAC with 8051 and generate different analog waveforms.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Maintain microcontroller based applications.

IV Laboratory Learning Outcome(s)

Interface DAC with 8051 microcontroller and generate triangular, saw-tooth waveform.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

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.

1. Binary weighted method and
2. R/2R ladder method.

R/2R ladder is the widely used method. This method can achieve a much higher 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. Number of data inputs decides the resolution of DAC. So, if there are n digital input pin, there are 2^n analog levels. So, 8 input DAC has 256 discrete voltage levels.

The output current is known as I_{out} by connecting a resistor to the output to convert into voltage. The total current provided by the I_{out} pin is basically a function of the binary numbers at the input pins D0 - D7 (D0 is the LSB and D7 is the MSB) of DAC0808 and the reference current I_{ref} . The following formula is showing the function of I_{out} :

$$I_{out} = I_{ref} \left(\frac{D_7}{2} + \frac{D_6}{4} + \frac{D_5}{8} + \frac{D_4}{16} + \frac{D_3}{32} + \frac{D_2}{64} + \frac{D_1}{128} + \frac{D_0}{256} \right)$$

The I_{ref} is the input current. This must be provided into the pin 14. Generally, 2.0mA is used as I_{ref} . I_{out} pin is connected to the resistor to convert the current to voltage.

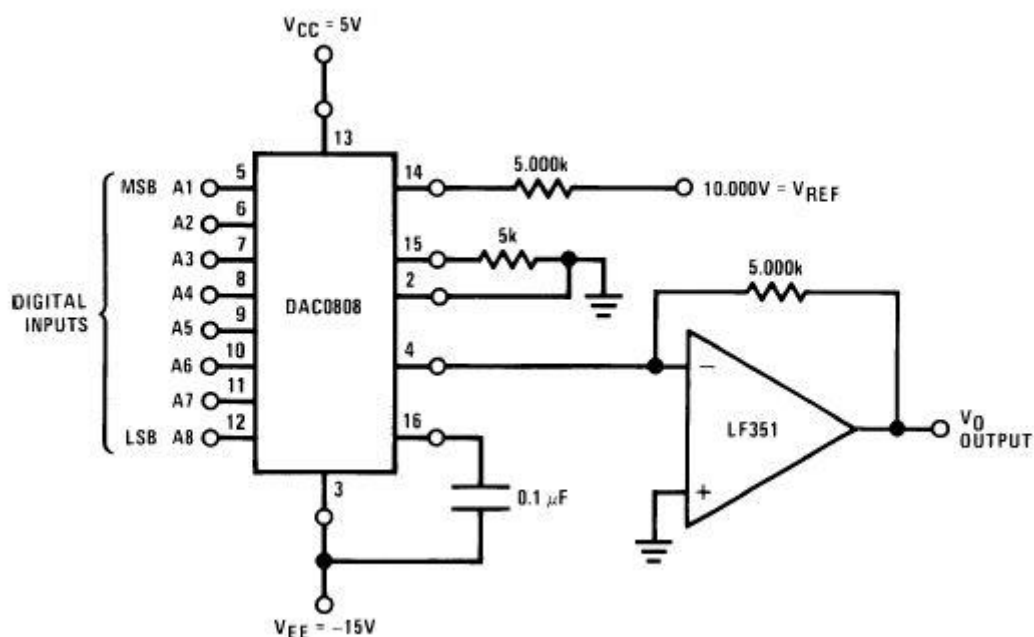


Fig 24.1: DAC Block Diagram

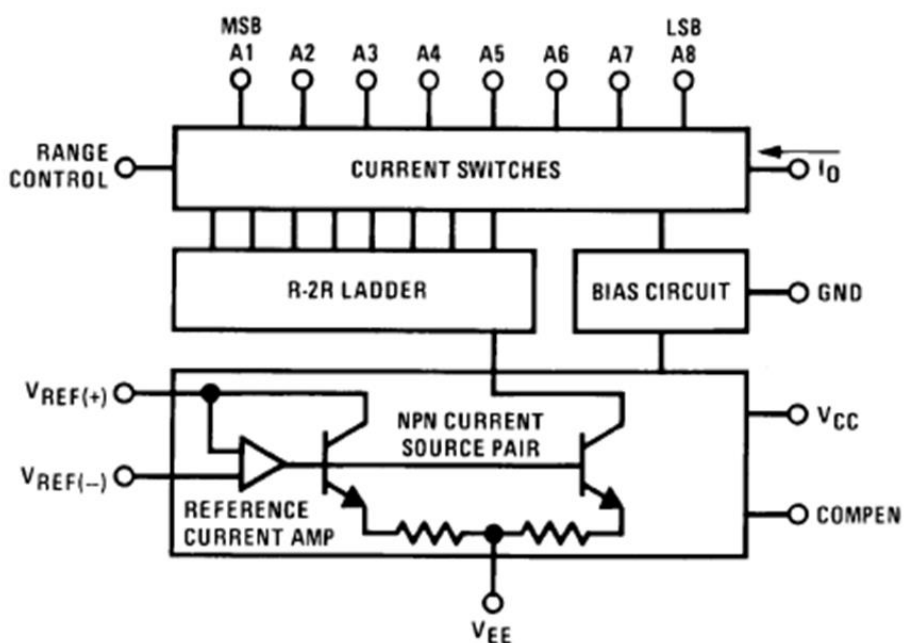


Fig 24.2: DAC Block Diagram

Specifications DAC 0808 Chip

1. Resolution: 8 bits
2. Settling Time: Typically, 100 nanoseconds
3. Output Current: Up to 1.999 mA
4. Voltage Supply: +5V or ±5V to ±18V
5. Accuracy: ±0.19% of full scale
6. Output Type: Voltage output
7. Compatibility: TTL/CMOS

VII Actual Circuit Diagram used in laboratory

a) Sample Circuit Diagram

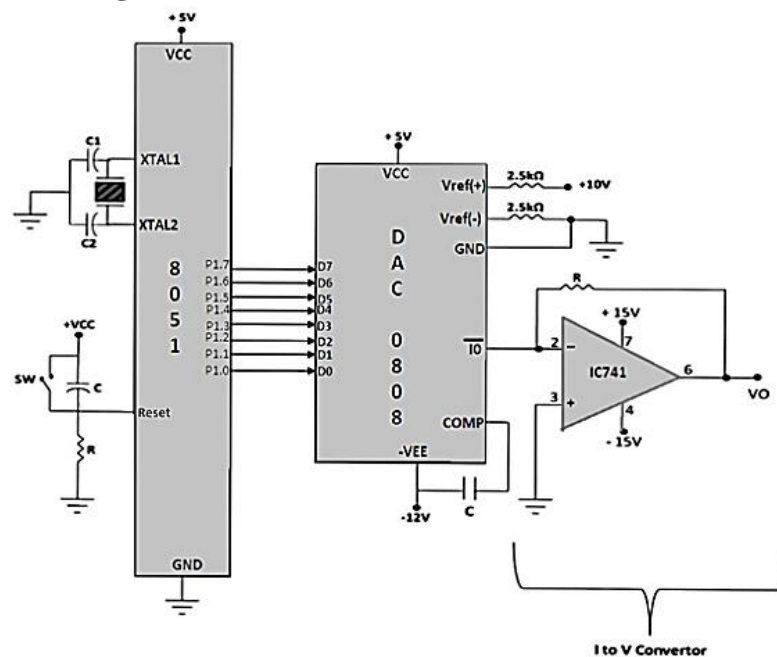


Fig 24.3 DAC interfacing to 8051

b) Practical Setup:

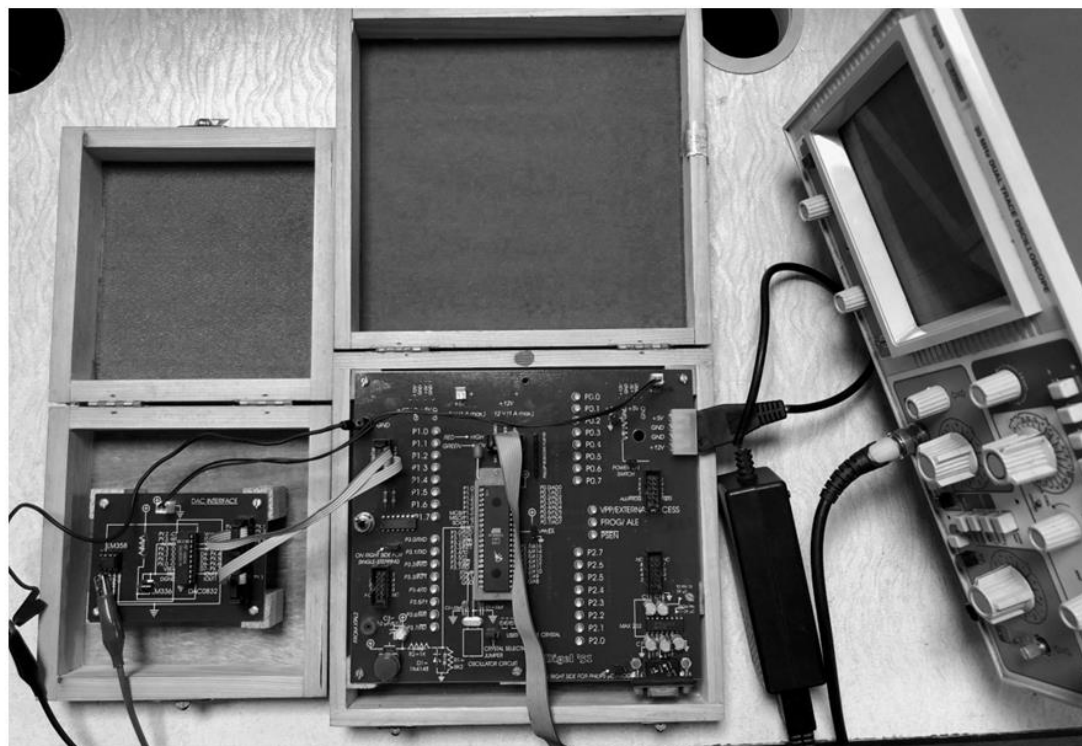


Fig 24.4 Practical Set up Diagram

c) **Simulation Diagram:**

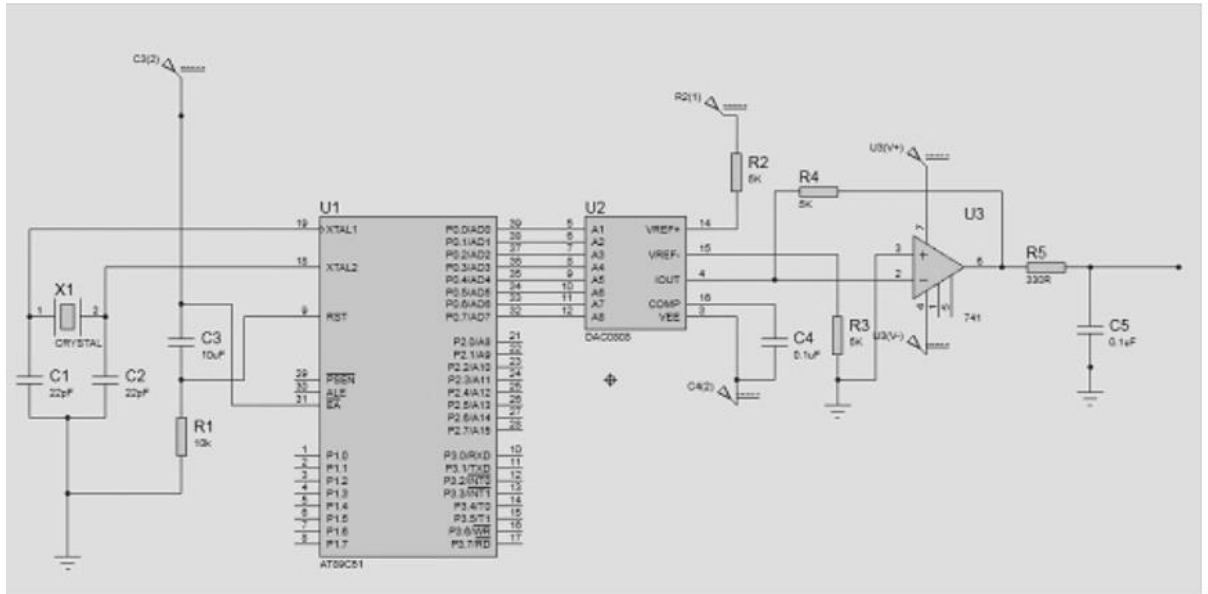


Fig 24.5: Simulation Diagram

d) **Actual Circuit Diagram used:**

VIII Resources Required

Sr. No.	Instrument /Components	Specification	Quantity
1	Microcontroller kit	Single board systems with 8K RAM, ROM memory with battery backup, 16X4, 16X2 LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross c-compiler, RS-232, USB, interfacing facility with built in power supply.	1 No.
2	Desktop PC	Loaded with open-source IDE, simulation and program downloading software	1 No.
3	DAC (0808) trainer board	Suitable to interface 8051 board.	1 No

IX Precautions to be Followed

1. Check rules / syntax of assembly programming.
2. Operate DAC chip as per specifications given in the datasheet otherwise damage may occur to the device.

X Procedure**Write Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write a program to generate triangular waveform using DAC.

STEP1- Algorithm

1. Make the Port used to Interface DAC as an output port.
2. Clear Accumulator.
3. Send 00H value to DAC
4. Increment value.
5. Compare current value with highest value 0FFh and send it to DAC till it reaches.
6. Decrement value.
7. Compare current value with lowest value 00H and send it to DAC till it reaches.
8. For repeat operation go to step3.
9. Stop.

Step 2-Flow Chart

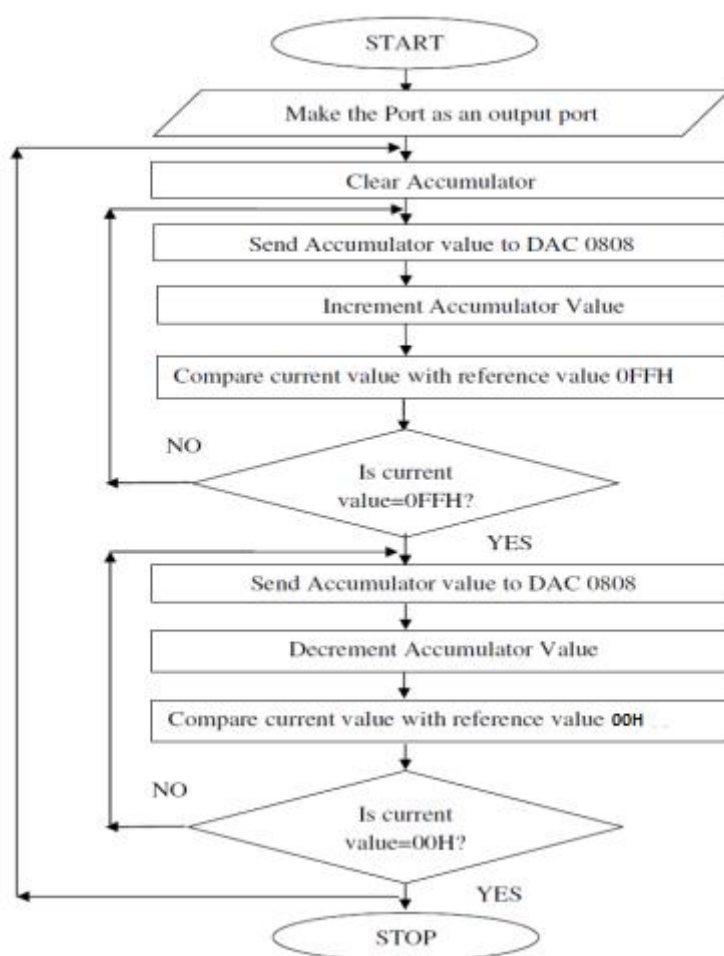


Fig 24.6 Flowchart to generate triangular waveform

Step 3-Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000h	
C:0x0000	7400	REPEAT:	MOV A, #00h	Clear A
C:0x0002	F590	INCR:	MOV P0, A	Send value to P1
C:0x0004	04		INC A	Increment value
C:0x0005	B4FFFA		CJNE A,#0FFh,INCR	Compare with highest value
C:0x0008	F590	DECR:	MOV P0, A	Send value to P1
C:0x000A	14		DEC A	Decrement value
C:0x000B	B400FA		CJNE A, #00h, DECR	Compare with lowest value
C:0x000E	80F0		SJMP REPEAT	Repeat
			END	

Problem statement for student: Develop assembly program to generate sawtooth waveform using DAC 0808

Step 1-Algorithm	Step 2-Flowchart

XIII Observations for sample program (use blank sheet provided if space not sufficient)
Trace the waveform for the triangular waveform observed on CRO.

XIV Results (Output of the Program)

.....
.....
.....
.....

XV Interpretation of Results (Give meaning of the above obtained results)

.....
.....
.....
.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....
.....
.....
.....

XVII Practical Related Questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identified CO

1. If $I_{ref} = 2\text{mA}$ and all the inputs to the DAC are high then find maximum current of DAC 0808 IC.
2. To generate a sine wave using DAC 0808 find decimal values representing magnitude of the sine of angles between 0 and 360 degrees.
Refer $V_{out} = 5V + (5x\sin\theta)$.

3. <https://peripheralinterfacing.wordpress.com/digital-to-analog-converter-using-8051-microcontroller/>

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40% (10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 % (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 25: Stepper Motor interfacing to 8051

I Practical Significance

Stepper motors are commonly used in a wide range of applications such as robotics, CNC machines, 3D printers, medical equipment, and automated manufacturing machinery where precise speed control and accuracy are necessary. Stepper motors are controlled by microcontrollers in areas such as computer peripherals, Business machines, process control and for making robots. This practical will help the students to develop skills to interface stepper motor to 8051 and rotate in clockwise direction at the given angles

II Industry/Employer Expected Outcome(s)

Maintain microcontroller based systems.

III Course Level Learning Outcome(s)

Maintain microcontroller based applications

IV Laboratory Learning Outcome(s)

Interface stepper motor to microcontroller and rotate in clockwise direction at the given angles

V Relevant Affective Domain related outcome(s)

1. Follow safe practices.
2. Maintain tools and equipment.
3. Follow ethical practices.

VI Relevant Theoretical Background

Stepper motor converts electrical energy into precise mechanical motion. It rotates a specific incremental distance per step. Every discrete step of a stepper motor is measured in degree (it can be 1.8° or even smaller depending on the stepper motor type and stepping technique).

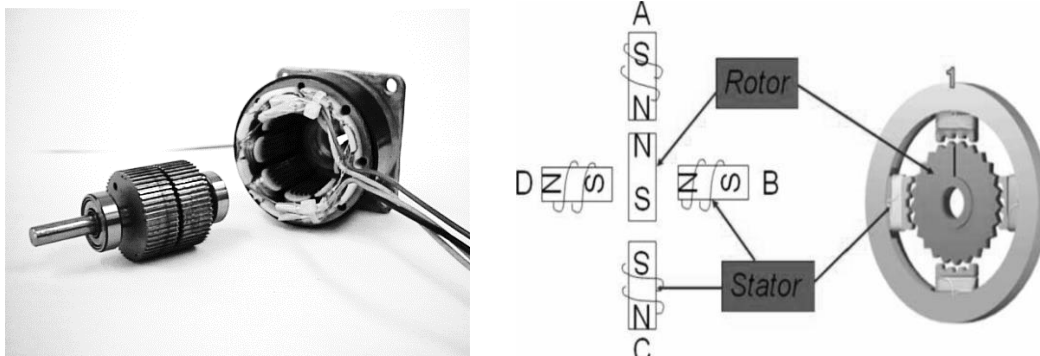


Fig.25.1 Construction of stepper motor

It works on the principle of electromagnetism. There is a magnetic rotor shaft of soft iron which is surrounded by the electromagnetic stators. When the stators are energized the rotor moves to align itself along with the stator (in case of a permanent magnet type stepper) or moves to have a minimum gap with the stator (in case of variable reluctance stepper). In this way the stators are energized in a sequence to rotate stepper motor

Generally, stepper motors are classified into two types according to windings as:

- Unipolar
- Bipolar

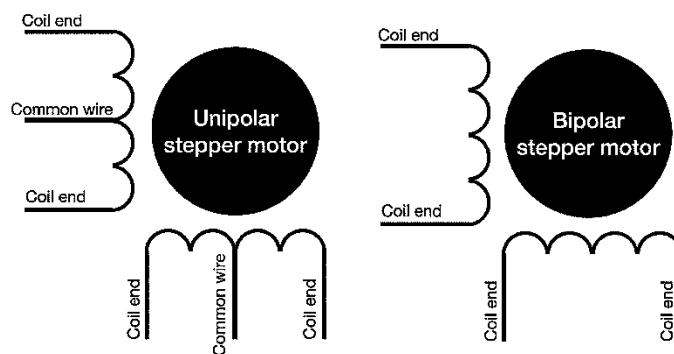


Fig.25.2 Stepper motor types

The unipolar stepper motors, has one winding per phase, with a center tap. It has five or six wires and four coils (actually two coils divided by center connections on each coil). The centre connections of the coils are tied together and used as the power connection. They are called unipolar stepper motors because power always comes in one pole.

The Bipolar stepper motor has four wires with no common center connection. It has two independent sets of coils.

Step angle:

Step angle is defined as the minimum degree of rotation with a single step.

Number of steps per revolution = $360^\circ / \text{step angle}$

Steps per second = $(\text{RPM} \times \text{steps per revolution}) / 60$

Table 18.1 Stepper Motor Step Angles

Step Angle	Steps per revolution
0.72	500
1.8	200
2.0	180
2.5	144
5	72

Switching Sequence of Motor:

As the coils need to be energized for the rotation. This can be done by sending a bits sequence to one end of the coil while the other end is commonly connected. The bit sequence sent can make either one phase ON or two phase ON for a full step sequence or it can be a combination of one and two phase ON for half step sequence.

Full Step Sequence:

a) Wave Drive Stepping Mode (ONE PHASE ON)

Clockwise	Step #	Winding A	Winding B	Winding C	Winding D	Anti-Clockwise
↓	1	1	0	0	0	↑
	2	0	1	0	0	
	3	0	0	1	0	
	4	0	0	0	1	

b. Full Drive Stepping Mode (TWO PHASE ON)

Clockwise	Step #	Winding A	Winding B	Winding C	Winding D	Anti-Clockwise
↓	1	1	0	0	1	↑
	2	1	1	0	0	
	3	0	1	1	0	
	4	0	0	1	1	

Half Step Sequence:

- Half Drive mode:

To allow for finer resolution, all stepper motors allow an 8-step switching sequence.

Clockwise	Step	Winding A	Winding B	Winding C	Winding D	Clockwise
↓	1	1	0	0	1	↑
	2	1	0	0	0	
	3	1	1	0	0	
	4	0	1	0	0	
	5	0	1	1	0	
	6	0	0	1	0	
	7	0	0	1	1	
	8	0	0	0	1	

VII Practical Circuit diagram:

a) Sample circuit diagram

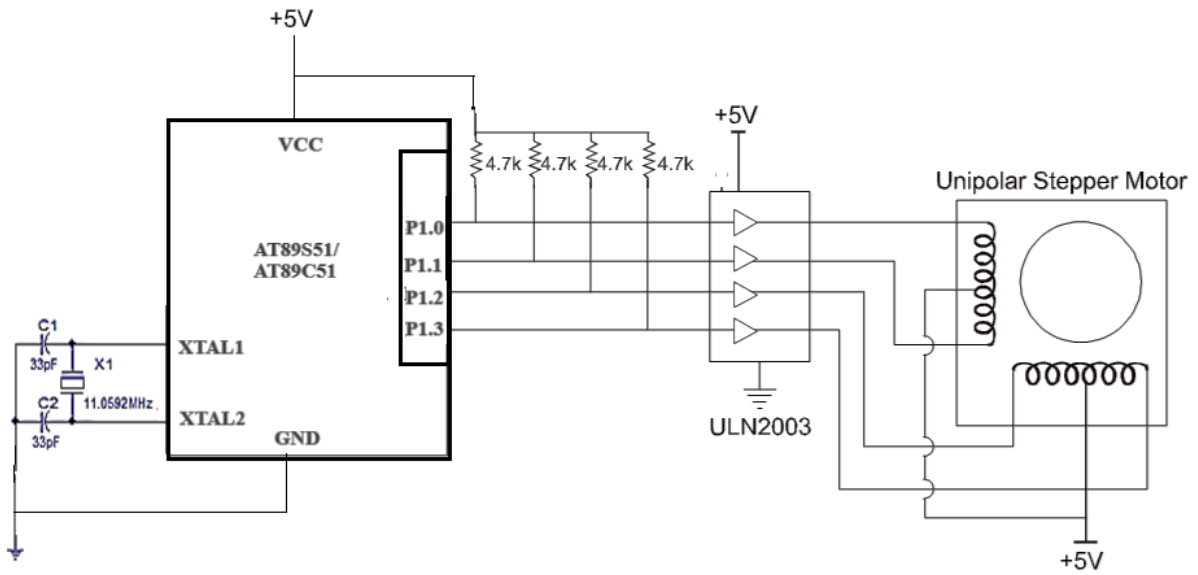


Fig 25.3 8051 connection to stepper motor

b) Practical setup

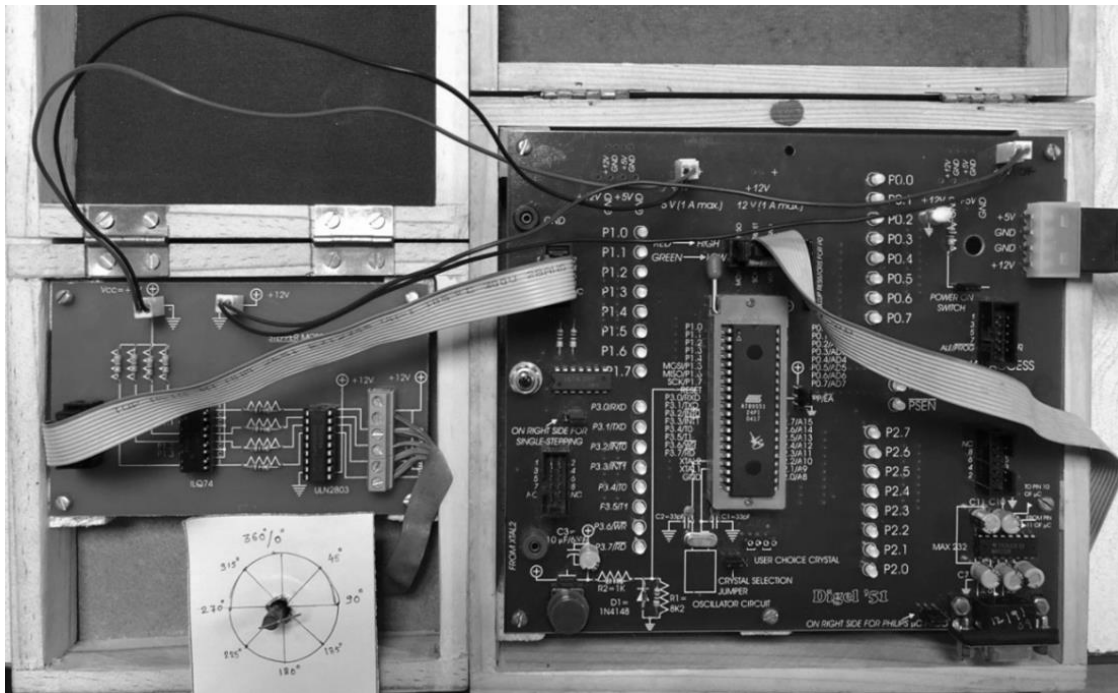


Fig 25.4 Practical Setup

c) Simulation diagram

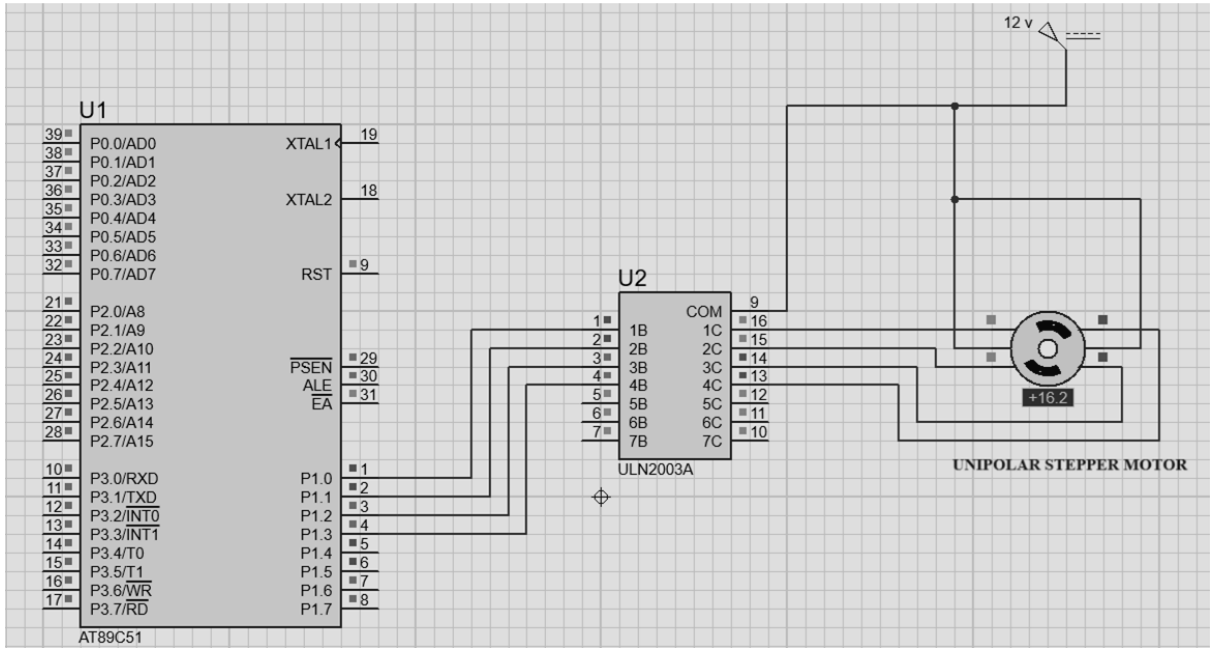


Fig 25.5 Simulation diagram

d) Actual circuit used in laboratory

- e) Actual Experimental set up used in laboratory

VIII Required Resources/apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1.	Microcontroller kit	Single board system with 8K RAM,ROM memory with battery backup,16X4,16X2LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross C-compiler,RS-232,USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open source IDE, simulation and program downloading software.	1 No.
3.	Stepper Motor Trainer	1.8° Step angle, 50/100 RPM Stepper motor with ULN 2003/2803 Driver.	No.

IX Precautions to be followed

- 1) Ensure proper connection before turning ON power supply to the kit.
- 2) Always use driver circuit while interfacing stepper motor to microcontroller.
- 3) Check rules / syntax of assembly language programming.

X Procedure

1. Write algorithm for given problem.
2. Draw flowchart for the same.
3. Develop assembly program using Integrated Development Environment (IDE) or any other relevant software tool.
4. Debug program on IDE.
5. Execute program on IDE.
6. Create hex file for the above program.
7. Download hex code in EPROM/Flash memory of the microcontroller.
8. Interface stepper motor to microcontroller as per circuit diagram shown in fig 25.3
9. Observe rotation of stepper motor and record in observation Table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM: Write a program to rotate stepper motor in clockwise direction by 360°
 Assume step angle of 1.8°

Step 1-Algorithm

1. Initialize the Port used to Interface stepper motor as an output port.
2. Set register as counter R1=50 for 200 steps i.e. 360° rotation.
3. Send first step sequence to stepper motor and add delay
4. Send second step sequence to stepper motor and add delay
5. Send third step sequence to stepper motor and add delay
6. Send fourth step sequence to stepper motor and add delay
7. Decrement counter and if count $\neq 0$, go to step 3
8. Stop

Step 2-Flow Chart

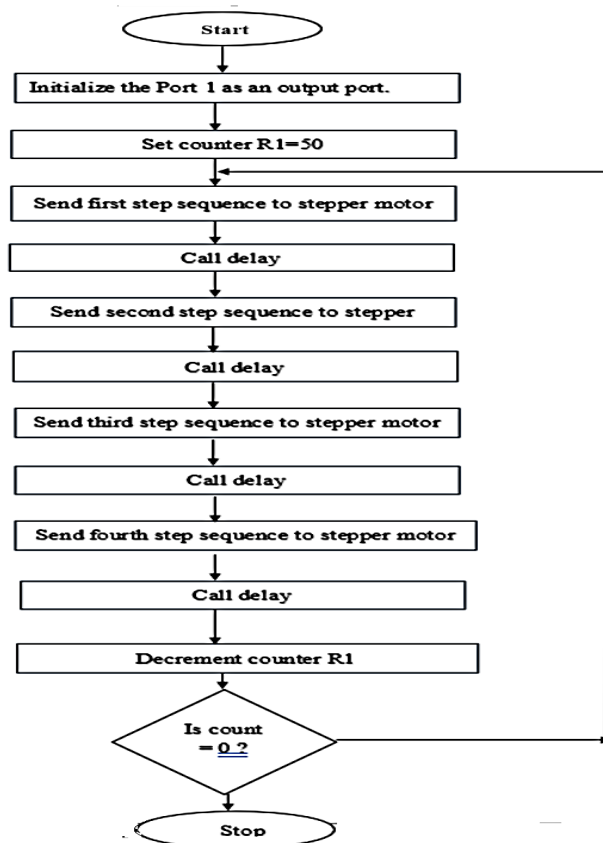


Fig 25.6 Flowchart for stepper motor to rotate in clockwise direction by 360°

Step 3-Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0x0000	7932		MOV R1,#50	; Load count =50 for 360 ⁰ rotation
C:0x0002	759099	UP:	MOV P1,#99H	; Send first step sequence to Port 1
C:0x0005	111A		ACALL DELAY	; Call delay subroutine
C:0x0007	7590CC		MOV P1,#0CCH	; Send second step sequence to Port 1
C:0x000A	111A		ACALL DELAY	; Call delay subroutine
C:0x000C	759066		MOV P1,#66H	; Send third step sequence to Port 1
C:0x000F	111A		ACALL DELAY	; Call delay subroutine
C:0x0011	759033		MOV P1,#33H	; Send fourth step sequence to Port 1
C:0x0014	111A		ACALL DELAY	; Call delay subroutine
C:0x0016	D9EA		DJNZ R1,UP	;Repeat 50 times
C:0x0018	80FE	HERE:	SJMP HERE	;stop
C:0x001A	7BFF	DELAY:	MOV R3,#255	;Delay subroutine
C:0x001C	7CFF	L2:	MOV R4,#255	
C:0x001E	DCFE	L1:	DJNZ R4,L1	
C:0x0020	DBFA		DJNZ R3,L2	
C:0x0022	22		RET	
			END	

Problem statement for student: Write a program to rotate stepper motor in clockwise direction by 180° . Use Full step sequence

Step 1-Algorithm	Step 2-Flowchart

XIII Observations sample program (use blank sheet provided if space not sufficient)

Steps	Step code	Port pin status (0 / 1)			
		P1.3	P1.2	P1.1	P1.0
Step 1					
Step 2					
Step 3					
Step 4					

XIV Results (Output of the Program)

.....

XV Interpretation of Results (Give meaning of the above obtained results)

.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....

XVII Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifies CO.

1. Calculate no of steps to move stepper motor by 40 degree in clockwise direction if step angle is 2 degree, steps per revolution =180.
2. A stepper motor has a step angle of 7.2 degrees. Calculate number of steps per revolution.
3. Give the 8-step sequence of a stepper motor if code start with 0010.

[Space for Answers]

.....

XVIII References/Suggestions for further reading

1. <https://www.monolithicpower.com/learning/resources/stepper-motors-basics-types-uses>
2. <https://www.portescap.com/en/products/stepper-motor-control>
3. <http://vlabs.iitkgp.ernet.in/rtes/exp10/index.html#>
4. <https://www.tutorialspoint.com/interfacing-stepper-motor-with-8051microcontroller>

XIX Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 26: Stepper Motor interfacing to 8051 for rotating anticlockwise

I Practical Significance

Stepper motors are commonly used in a wide range of applications such as robotics, CNC machines, 3D printers, medical equipment, and automated manufacturing machinery where precise speed control and accuracy are necessary. Stepper motors are controlled by microcontrollers in areas such as computer peripherals, Business machines, process control and for making robots. This practical will help the students to develop skills to interface stepper motor to 8051 and rotate in clockwise direction at the given angles

II Industry/Employer Expected Outcome(s)

Maintain microcontroller based systems.

III Course Level Learning Outcome(s)

Maintain microcontroller based applications

IV Laboratory Learning Outcome(s)

Interface stepper motor to microcontroller and rotate in anti-clockwise direction at the given angles.

V Relevant Affective Domain related outcome(s)

1. Follow safe practices.
2. Maintain tools and equipment.
3. Follow ethical practices.

VI Relevant Theoretical Background

Stepper motor converts electrical energy into precise mechanical motion. It rotates a specific incremental distance per step. Every discrete step of a stepper motor is measured in degree (it can be 1.8° or even smaller depending on the stepper motor type and stepping technique).

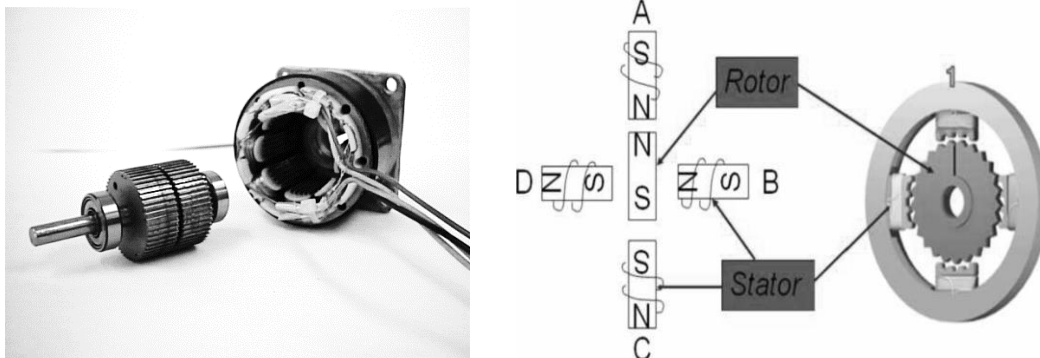


Fig.26.1 Construction of stepper motor

It works on the principle of electromagnetism. There is a magnetic rotor shaft of soft iron which is surrounded by the electromagnetic stators. When the stators are energized the rotor moves to align itself along with the stator (in case of a permanent magnet type stepper) or moves to have a minimum gap with the stator (in case of variable reluctance stepper). In this way the stators are energized in a sequence to rotate stepper motor

Generally, stepper motors are classified into two types according to windings as:

- Unipolar
- Bipolar

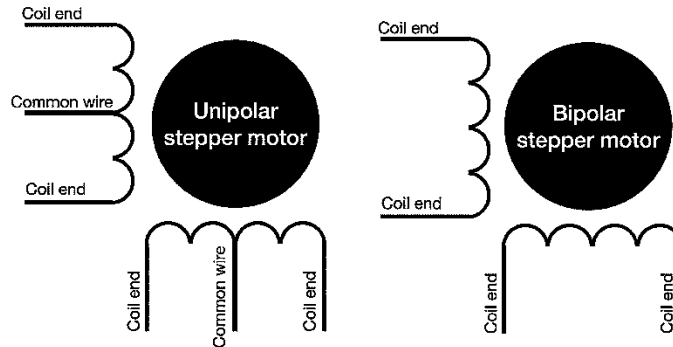


Fig.26.2 Stepper motor types

The unipolar stepper motors, has one winding per phase, with a center tap. It has five or six wires and four coils (actually two coils divided by center connections on each coil). The centre connections of the coils are tied together and used as the power connection. They are called unipolar stepper motors because power always comes in one pole.

The Bipolar stepper motor has four wires with no common center connection. It has two independent sets of coils.

Step angle:

Step angle is defined as the minimum degree of rotation with a single step.

Number of steps per revolution = $360^\circ / \text{step angle}$

Steps per second = $(\text{RPM} \times \text{steps per revolution}) / 60$

Table 26.1 Stepper Motor Step Angles

Step Angle	Steps per revolution
0.72	500
1.8	200
2.0	180
2.5	144
5	72

Switching Sequence of Motor:

As the coils need to be energized for the rotation. This can be done by sending a bits sequence to one end of the coil while the other end is commonly connected. The bit sequence sent can make either one phase ON or two phase ON for a full step sequence or it can be a combination of one and two phase ON for half step sequence.

Full Step Sequence:

a) Wave Drive Stepping Mode (ONE PHASE ON)

Table 26.2 Wave Drive Stepping Mode (ONE PHASE ON)

Clockwise	Step #	Winding A	Winding B	Winding C	Winding D	Anti-Clockwise
↓	1	1	0	0	0	↑
	2	0	1	0	0	
	3	0	0	1	0	
	4	0	0	0	1	

b. Full Drive Stepping Mode (TWO PHASE ON)

Table 26.3 Wave Drive Stepping Mode (ONE PHASE ON)

Clockwise	Step #	Winding A	Winding B	Winding C	Winding D	Anti-Clockwise
↓	1	1	0	0	1	↑
	2	1	1	0	0	
	3	0	1	1	0	
	4	0	0	1	1	

Half Step Sequence:

- Half Drive mode:

To allow for finer resolution, all stepper motors allow an 8-step switching sequence.

Table 26.4 8 Step Switching Sequence (ONE PHASE ON)

Clockwise	Step	Winding A	Winding B	Winding C	Winding D	Clockwise
↓	1	1	0	0	1	↑
	2	1	0	0	0	
	3	1	1	0	0	
	4	0	1	0	0	
	5	0	1	1	0	
	6	0	0	1	0	
	7	0	0	1	1	
	8	0	0	0	1	

VII Practical Circuit diagram:

a) Sample circuit diagram

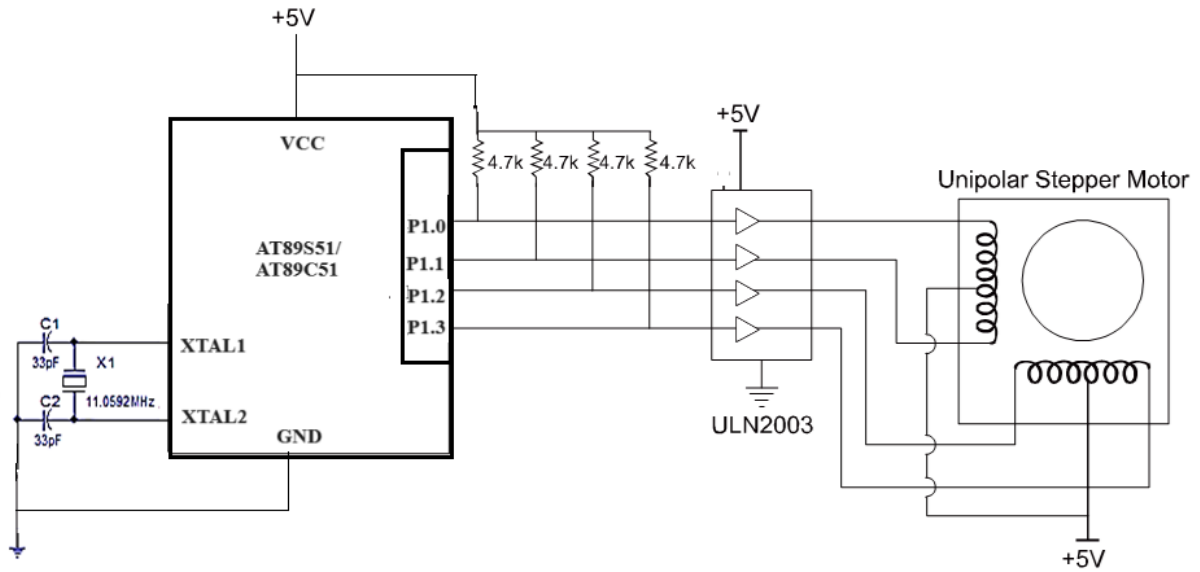


Fig 26.3 8051 connection to stepper motor

b) Practical setup

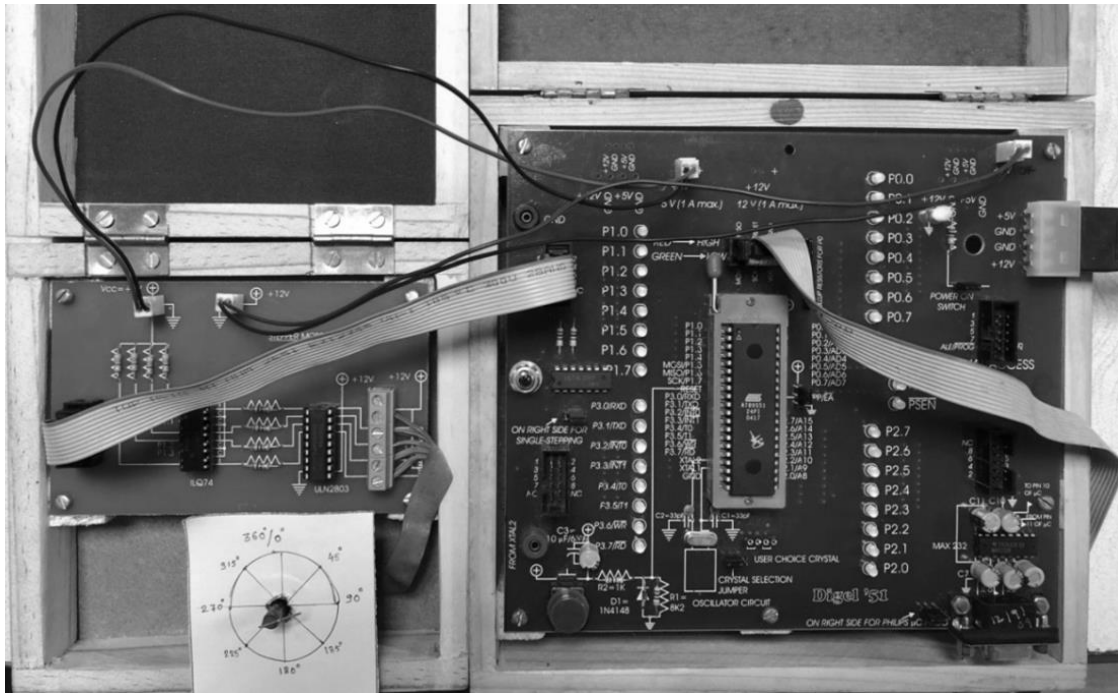


Fig 26.4 Practical Setup

c) Simulation diagram

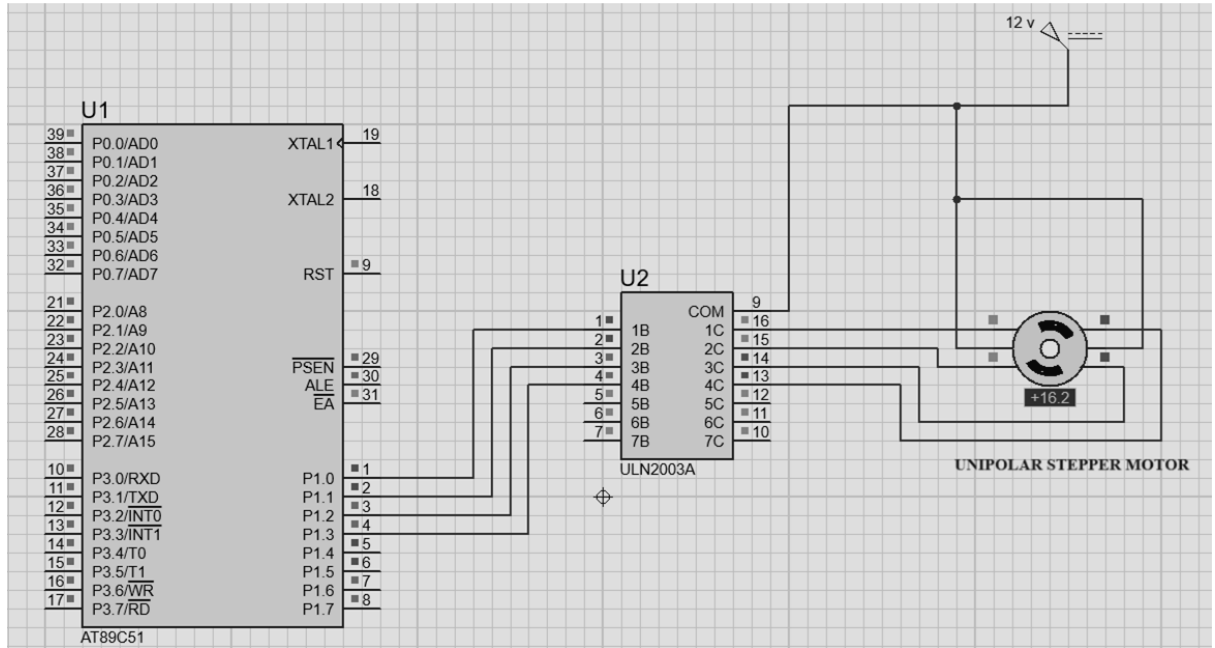


Fig 26.5 Simulation diagram

d) Actual circuit used in laboratory

e) Actual Experimental set up used in laboratory

VIII Required Resources/apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1.	Microcontroller kit	Single board system with 8K RAM, ROM memory with battery backup, 16X4, 16X2 LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross C-compiler, RS-232, USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open-source IDE, simulation and program downloading software.	1 No.
3.	Stepper Motor Trainer	1.8° Step angle, 50/100 RPM Stepper motor with ULN 2003/2803 Driver.	No.

IX Precautions to be followed

- 1) Ensure proper connection before turning ON power supply to the kit.
- 2) Always use driver circuit while interfacing stepper motor to microcontroller.
- 3) Check rules / syntax of assembly language programming.

X Procedure

1. Write algorithm for given problem.
2. Draw flowchart for the same.
3. Develop assembly program using Integrated Development Environment (IDE) or any other relevant software tool.
4. Debug program on IDE.
5. Execute program on IDE.
6. Create hex file for the above program.
7. Download hex code in EPROM/Flash memory of the microcontroller.
8. Interface stepper motor to microcontroller as per circuit diagram shown in fig 26.3
9. Observe rotation of stepper motor and record in observation Table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM: Write a program to rotate stepper motor in anti-clockwise direction by 180°
 Assume step angle of 1.8° Use half step sequence

Step 1-Algorithm

1. Make the Port used to Interface stepper motor as an output port.
2. Set register as counter R2 = 25 for 100 steps i.e. 180° rotation.
3. Set register as counter R3 = 8 for half step code
4. Initialize pointer to table which is in code memory i.e. DPTR.
5. Clear accumulator.
6. Read data from code memory.
7. Send code to stepper motor.
8. Increment DPTR to access next memory location code.
9. Decrement R3 and if $R3 \neq 0$, go to step 5 else go to next.
10. Decrement R2 and if $R2 \neq 0$, go to step 3 else go to next.
11. Stop.

Step 2-Flow Chart

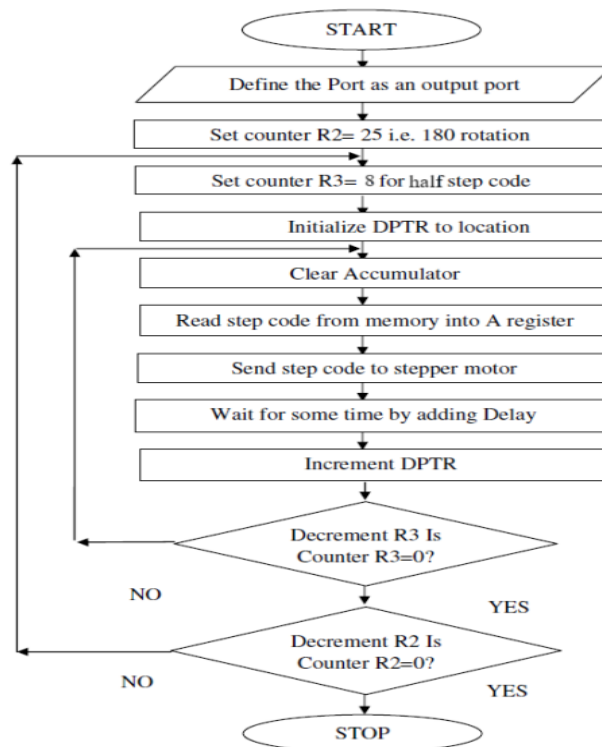


Fig 26.6 Flowchart for stepper motor to rotate in anti-clockwise direction by 180°

Step 3-Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0x0000	759000		MOV P1,#00H	;Define port P1 as output port
C:0x0003	7A19		MOV R2,#25	;Set register as counter of 25 for 180 ⁰ rotation
C:0x0005	7B08	UP1:	MOV R3,#8	;set counter of 8 for half step code sequence
C:0x0007	900200		MOV DPTR,#TABLE	;load address of program memory into Data pointer
C:0x000A	E4	UP:	CLR A	;clear accumulator
C:0x000B	93		MOVC A,@A+DPTR	;read step code from memory into accumulator
C:0x000C	F590		MOV P1,A	;send step code to port
C:0x000E	1117		ACALL DELAY	;add delay
C:0x0010	A3		INC DPTR	;increment memory pointer to read next step code
C:0x0011	DBF7		DJNZ R3,UP	; decrement counter & jump to memory location labeled as UP if not equal to zero.
C:0x0013	DAF0		DJNZ R2,UP1	; decrement counter & jump to memory location labeled as UP1 if not equal to zero.
C:0x0015	80FE	HERE:	SJMP HERE	;wait
C:0x0017	7C64	DELAY:	MOV R4,#100	;delay Subroutine
C:0x0019	7D19	L3:	MOV R5,#25	
C:0x001B	7E19	L2:	MOV R6,#25	
C:0x001D	DEFE	L1:	DJNZ R6,L1	
C:0x001F	DDFA		DJNZ R5,L2	
C:0x0021	DCF6		DJNZ R4,L3	
C:0x0023	22		RET	
			ORG 0050H	
C:0x0050		TABLE:	DB 01H,03H,02H, 06H, 04H, 0CH,08H, 09H,	; Step code stored at code memory starting at location 0050H onward.
			END	

Problem statement for student: Write a program to rotate stepper motor in anti-clockwise direction by 360° . Use full step sequence. Assume step angle of 1.8°

Step 1-Algorithm	Step 2-Flowchart

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

1.
2.
3.
4.
5.
6.
7.
8.

XIII Observations for sample Program (use blank sheet provided if space not sufficient)

Steps	Step code	Port pin status (0 / 1)			
		P1.3	P1.2	P1.1	P1.0
Step 1					
Step 2					
Step 3					
Step 4					
Step 5					
Step 6					
Step 7					
Step 8					

XIV Results (Output of the Program)

.....

XV Interpretation of Results (Give meaning of the above obtained results)

.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....
.....

XVII Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifies CO.

1. If a stepper motor takes 90 steps to make one complete revolution, calculate the step angle for this motor.
2. Write specifications of stepper motor used in the lab.

[Space for Answers]

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

XVIII References/Suggestions for further reading

1. <https://www.monolithicpower.com/learning/resources/stepper-motors-basics-types-uses>
2. <https://www.portescap.com/en/products/stepper-motor-control>
3. <http://vlabs.iitkgp.ernet.in/rtes/exp10/index.html#>
4. <https://www.tutorialspoint.com/interfacing-stepper-motor-with-8051microcontroller>

XIX Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 27: Water Level Controller using 8051

I Practical Significance

The control of water levels is critical in various environmental and industrial contexts. Water level control is essential for flood control, water supply management, ecosystem stability, energy production, climate resilience, infrastructure protection, and navigational safety. This practical will help the students to develop water level controller using 8051 and display the status of Water tank on LCD.

II Industry/Employer expected outcome(s)

Maintain microcontroller-based systems.

III Course Level Learning Outcome(s)

Maintain microcontroller-based applications.

IV Laboratory Learning Outcome(s)

Design water level controller using any suitable open-source simulation software to detect and control the water level in a tank.

V Relevant Affective domain related Outcome(s)

Follow ethical practices.

VI Relevant Theoretical Background

Sensors are crucial for measuring water levels accurately. Ultrasonic, capacitive, and float sensors provide the necessary data for system adjustments. Advanced control systems utilize automation and robotics to operate gates, pumps, and valves, ensuring optimal water levels without human intervention. Thus, the water level controllers integrate control theory, sensor technology, hydraulic modeling, automation, mechanical engineering, communication technologies, and feedback mechanisms to maintain desired water levels efficiently.

Water level control system utilizes sensors to determine the water level in the tank which provides the real time value to the microcontroller 8051 which compares with the set value and takes decision whether to turn off the motor or keep it on which controls the water level in the tank and also displays the status of water level on LCD display.

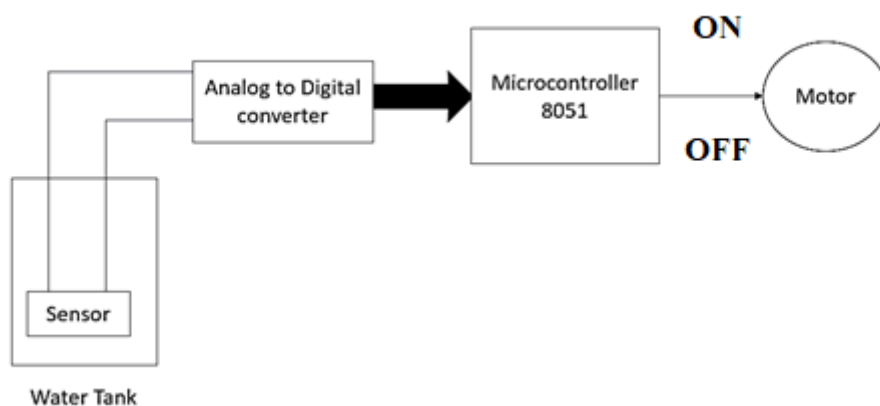


Fig 27.1 Water Level Controller basic block diagram

VII Actual Circuit Diagram used in laboratory

a) Simulation Diagram:

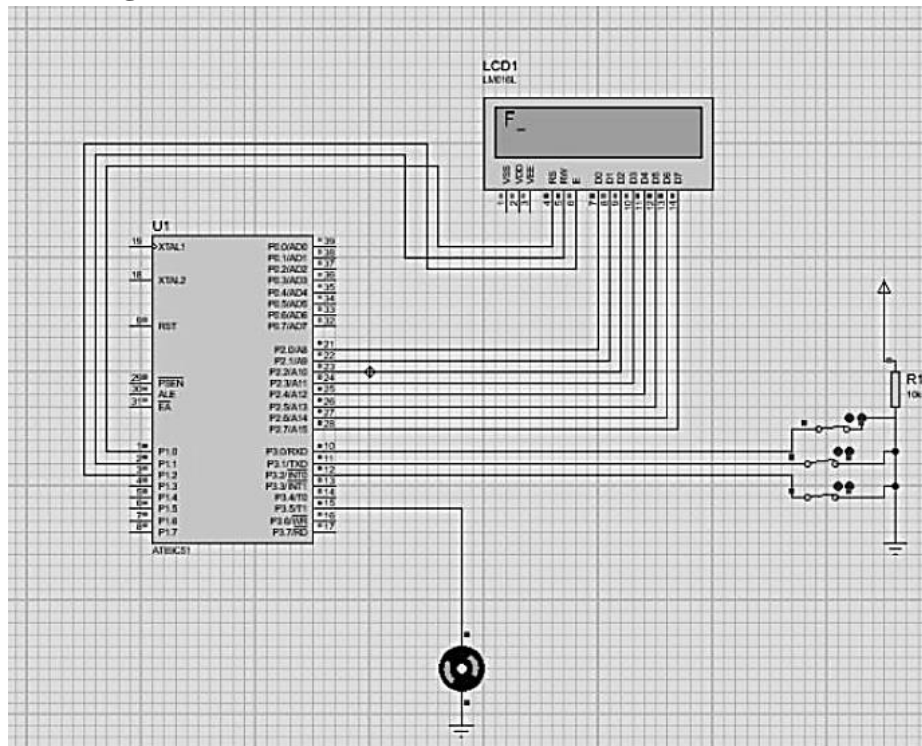


Fig 27.1: Simulation Diagram

b) Simulation Diagram used:

VIII Resources Required

Sr. No.	Instrument /Components	Specification	Quantity
1	Microcontroller kit	Single board systems with 8K RAM, ROM memory with battery backup, 16X4, 16X2 LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross c-compiler, RS-232, USB, interfacing facility with built in power supply.	1 No.
2	Desktop PC	Loaded with open-source IDE, simulation and program downloading software	1 No.
3	Resistor	10 Kilo Ohms	1No.
4.	Liquid Crystal Display	16 x 2 LCD	1 No.
5.	Simple DC Motor	Nominal Voltage 12V, Load Resistance 12 Ohms	1No.

IX Precautions to be Followed

1. Check rules / syntax of assembly programming.
2. Operate DAC chip as per specifications given in the datasheet otherwise damage may occur to the device.

X Procedure**Write Program**

1. Start Keil by double clicking on Keil icon.
2. Create a new project.
3. Select device for Target.
4. Double click on ATMEL and select AT89C51.
5. Type the program in text editor and save as filename.asm extension.

Compile the Program

6. Right click on source group and build the target.
7. Check for any errors in the output window and remove if any.

Run, Debug the Program

8. Click on Debug and start simulation and start/stop debug session.
9. Run the program step by step.
10. Observe the output on the project window.
11. Note the values of the result of various operations in the observation table.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM 1: Write a program to detect and control the water level in tank and display the status on LCD.

STEP1- Algorithm

1. Configure the Port 0 and Port 2 as an output port.
2. Configure Port 3 pin 5 as output port pin to make the motor ON and OFF.
3. Configure Port 3 pin 3.0 to pin 3.3 as input pins to sense the status of Water level in tank.
4. Determine the water level by reading status of sensors connected.
5. If water tank is full then turn off the motor and display full on LCD.
6. Else keep the motor ON and display the status on LCD.
7. Repeat the process by going back to step 4.

Step 2-Flow Chart

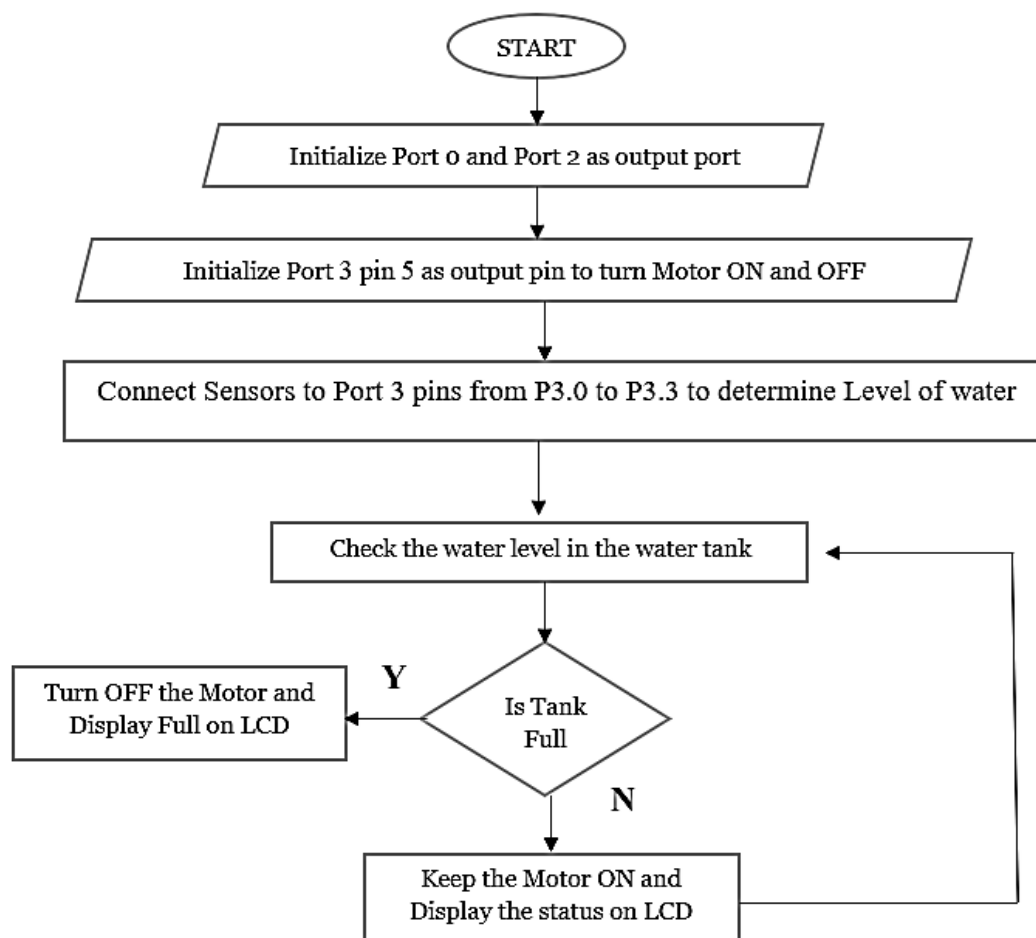


Fig 27.2 Flowchart for Water Level Controller

Step 3-Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			RS BIT P1.0	
			RW BIT P1.1	
			E BIT P1.2	

Memory Address	Hex Code	Label	Mnemonics	Comments
			ORG 0000H	
C:0X0000	30B00E	CHECK:	JNB P3.0, LCDDISPLAY	Check the status of water level
C:0X0003	30B10B		JNB P3.1, LCDDISPLAY	Check the status of water level
C:0X0006	C285		CLR P3.5	
C:0X0008	30B206		JNB P3.2, LCD DISPLAY	Check the status of water level
C:0X000B	D2B5		SETB P3.5	
C:0X000D	1111		ACALL LCD DISPLAY	
C:0X000F	80EF		SJMP CHECK	
C:0X0011	C290	LCDDISPLAY:	CLR RS	
C:0X0013	C291		CLR RW	
C:0X0015	C292		CLR E	
C:0X0017	7438		MOV A, #38H	Initialize LCD
C:0X0019	1162		ACALL LCDCMD	
C:0X001B	117C		ACALL DELAY	
C:0X001D	740E		MOV A, #0EH	
C:0X001F	1162		ACALL LCDCMD	
C:0X0021	117C		ACALL DELAY	
C:0X0023	7406		MOV A, #06H	
C:0X0025	1162		ACALL LCDCMD	
C:0X0027	117C		ACALL DELAY	
C:0X0029	7401		MOV A, #01H	
C:0X002B	1162		ACALL LCDCMD	
C:0X002D	117C		ACALL DELAY	
C:0X002F	7480		MOV A, #80H	
C:0X0031	1162		ACALL LCDCMD	
C:0X0033	117C		ACALL DELAY	
C:0X0035	30B008		JNB P3.0, F_DISPLAY	
C:0X0038	30B10F		JNB P3.1, H_DISPLAY	
C:0X003B	30B214		JNB P3.2, Q_DISPLAY	
C:0X003E	115A		ACALL E_DISPLAY	
C:0X0040	7446	F_DISPLAY:	MOV A, # "F"	Display Tank Full
C:0X0042	1164		ACALL LCDDATA	Call Subroutine for LCD display
C:0X0044	117C		ACALL DELAY	Call Delay
C:0X0046	C2B5		CLR P3.5	
C:0X0048	80B6		SJMP CHECK	
C:0X004A	7448	H_DISPLAY:	MOV A, # "H"	Display Tank Half Full

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0X004C	116F		ACALL LCDDATA	Call Subroutine for LCD display
C:0X004E	117C		ACALL DELAY	Call Delay
C:0X0050	80AE		SJMP CHECK	
C:0X0052	7451	Q_DISPLAY:	MOV A, # "Q"	Display Tank Quarter Full
C:0X0054	116F		ACALL LCDDATA	Call Subroutine for LCD display
C:0X0056	117C		ACALL DELAY	Call Delay
C:0X0058	80A6		SJMP CHECK	
C:0X005A	7445	E_DISPLAY:	MOV A, # "E"	Display Tank Empty
C:0X005C	116F		ACALL LCDDATA	Call Subroutine for LCD display
C:0X005E	117C		ACALL DELAY	Call Delay
C:0X0060	809E		SJMP CHECK	
C:0X0062	C290	LCDCMD:	CLR RS	Subroutine for LCD Commands
C:0X0064	C291		CLR RW	
C:0X0066	F5A0		MOV P2, A	
C:0X0068	D292		SETB E	
C:0X006A	117C		ACALL DELAY	
C:0X006C	C292		CLR E	
C:0X006E	22		RET	
C:0X006F	D290	LCDDATA:	SETB RS	Subroutine for LCD data display
C:0X0071	C291		CLR RW	
C:0X0073	F5A0		MOV P2, A	
C:0X0075	D292		SETB E	
C:0X0077	117C		ACALL DELAY	
C:0X0079	C292		CLR E	
C:0X007B	22		RET	
C:0X007C	A8FF	DELAY:	MOV R0, #FFH	Delay Subroutine
C:0X007E	79FF	BACK1:	MOV R1, #FFH	
C:0X0080	D9FE	HERE:	DJNZ R1, HERE	
C:0X0082	D8FA		DJNZ R0, BACK1	
C:0X0084	22		RET	
			END	

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

XVIII References / Suggestions for further reading

1. <https://www.electronicshub.org/water-level-controller-using-8051-microcontroller/>
2. <https://www.circuitstoday.com/water-level-controller-using-8051>
3. <https://www.electronicsforu.com/electronics-projects/hardware-diy/water-level-controller-cum-motor-protector>

XIX Assessment Scheme

The given performance indicators should serve as a guideline for assessment regarding process and product related marks:

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40% (10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 % (25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	

Practical No. 28: Temperature Sensor interfacing to detect and measure temperature

I Practical Significance

The measurement of temperature by using appropriate sensors and controllers is not only important in environmental or weather monitoring but also crucial for many industrial processes. Usually, a temperature sensor converts the temperature into an equivalent voltage output. IC LM35 is such a sensor. This practical will help the students to develop skills to interface temperature sensor to the microcontroller, read temperature and display its value on LCD.

II Industry/Employer Expected Outcome(s)

Maintain microcontroller based systems.

III Course Level Learning Outcome(s)

Maintain microcontroller based applications

IV Laboratory Learning Outcome(s)

Interface temperature sensor LM35 to 8051 to read temperature, convert it to decimal and send the value to Port 0 with some delay

V Relevant Affective Domain related outcome(s)

1. Follow safe practices.
2. Maintain tools and equipment.
3. Follow ethical practices.

VI Relevant Theoretical Background

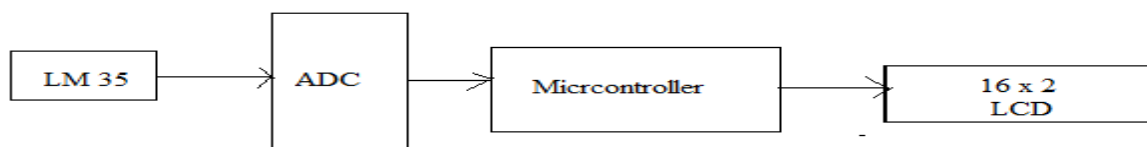


Fig.28.1 Temperature monitoring system

LM35 Temperature Sensor:

The LM35 series are precision integrated-circuit analog temperature sensor with an output voltage linearly proportional to the Centigrade temperature around it. It is a small and cheap IC which can be used to measure temperature anywhere between -55°C to 150°C . The LM35 comes already calibrated hence requires no external calibration.

LM35 Features:

1. Calibrated Directly in Celsius (Centigrade)
2. Operates From 4 V to 30 V. Typically 5V.
3. Can measure temperature ranging from -55°C to 150°C
4. Output voltage is directly proportional (Linear) to temperature
5. ±0.5°C Accuracy

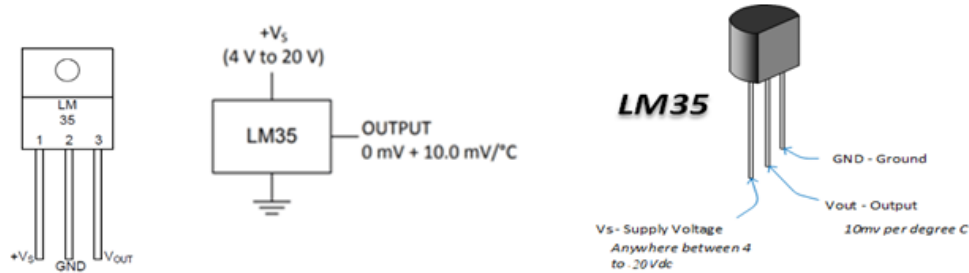


Fig.28.2 Pin Diagram of LM35

Pin description:

Pin Number	Pin Name	Description
1	Vs	Input voltage is +5V for typical applications
2	Analog Out	There will be increase in 10mV for raise of every 1°C.
3	Ground	Connected to ground of circuit

Temperature to voltage conversion:

$$V_{OUT} = 10\text{mV} / ^\circ\text{C} \times T$$

$$T = V_{OUT} / 10 \text{ mV(milliVolt)}$$

Where Vout = LM35 output voltage, T = Temperature sensed in centigrade

This means at 100°C, Vout = 10 mV x 100 = 1000 mV = 1Volt

ADC0808:

An analog-to-digital converter, or simply ADC, is a semiconductor device that is used to convert an analog signal into a digital code. An analog signal is a signal that may assume any value within a continuous range. Examples of analog signals commonly encountered every day are sound, light, temperature, and pressure, all of which may be represented electrically by an analog voltage or current.

Specifications ADC0808 Chip	ADC0808 Analog signal selection:																																				
1. Resolution -8 Bits																																					
2. Input Channels- 8																																					
3. Single Supply- 5VDC																																					
4. Low Power- 15mW																																					
5. Conversion Time -100µs																																					
6. The clock frequency range of ADC is 10 KHz to 1280 KHz.																																					
7. Typically 680 kHz used.																																					
8. Low power consumption																																					
	<table border="1"> <thead> <tr> <th>Analog Channel</th> <th>C</th> <th>B</th> <th>A</th> </tr> </thead> <tbody> <tr> <td>IN0</td> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>IN1</td> <td>0</td> <td>0</td> <td>1</td> </tr> <tr> <td>IN2</td> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>IN3</td> <td>0</td> <td>1</td> <td>1</td> </tr> <tr> <td>IN4</td> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>IN5</td> <td>1</td> <td>1</td> <td>1</td> </tr> <tr> <td>IN6</td> <td>1</td> <td>1</td> <td>0</td> </tr> <tr> <td>IN7</td> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	Analog Channel	C	B	A	IN0	0	0	0	IN1	0	0	1	IN2	0	1	0	IN3	0	1	1	IN4	1	0	0	IN5	1	1	1	IN6	1	1	0	IN7	1	1	1
Analog Channel	C	B	A																																		
IN0	0	0	0																																		
IN1	0	0	1																																		
IN2	0	1	0																																		
IN3	0	1	1																																		
IN4	1	0	0																																		
IN5	1	1	1																																		
IN6	1	1	0																																		
IN7	1	1	1																																		

ADC0808 Pin diagram:

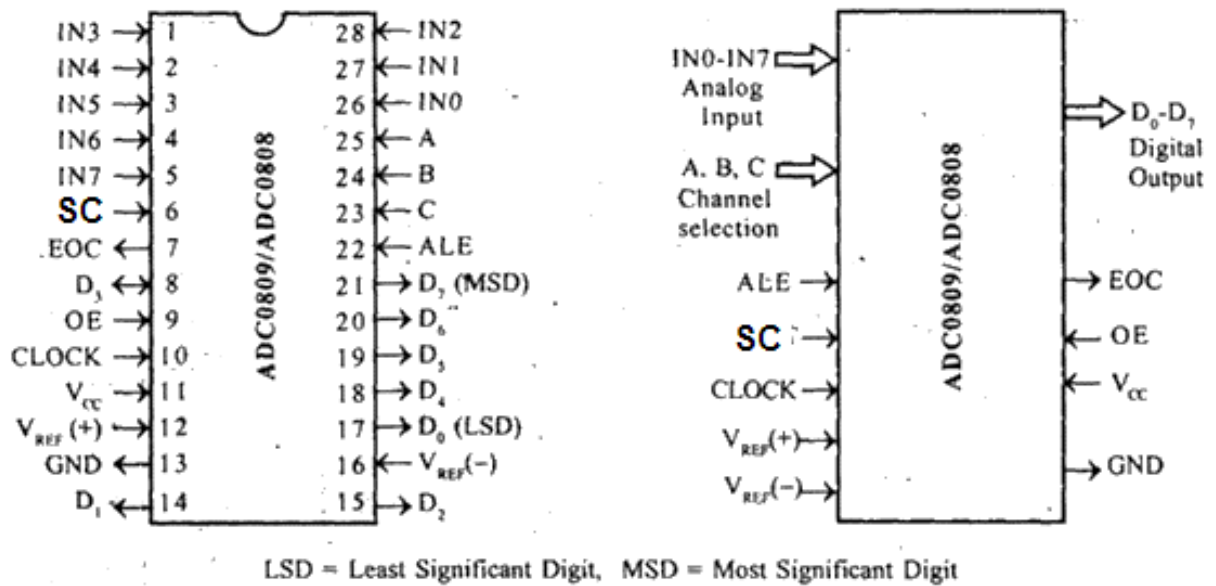


Fig 28.3 ADC 0808 Pin diagram

Table 28. 1: ADC 0808 Pin functions

Signals	Description
IN0-IN7	Eight single ended analog input to ADC.
A, B, C	3-bit binary input to select one of the eight analog signals for conversion at any one time.
ALE	Address latch enable. Used to latch the 3-bit address input to an internal latch.
SC	Start of conversion pulse input. To start ADC process this signal should be asserted high and then low . This signal should remain high for atleast 100ns.
CLOCK	Clock input and the frequency of clock can be in the range of 10 kHz to 1280 kHz. Typical clock input is 640 kHz.
$V_{REF}(+), V_{REF}(-)$	Reference voltage input. The positive reference voltage can be less than or equal to V_{cc} and the negative reference voltage can be greater than or equal to ground.
D_0-D_7	The 8-bit digital output. The reference voltages will decide the mapping of analog input to digital data.
EOC	End of conversion. This signal is asserted high by the ADC to indicate the end of conversion process and it can be used as interrupt signal to processor
OE	Output buffer Enable. This signal is used to read the digital data from output buffer after a valid EOC.
V_{cc}	Power supply, +5V
GND	Power supply ground, 0V

VII Practical Circuit diagram:

a) Sample circuit diagram

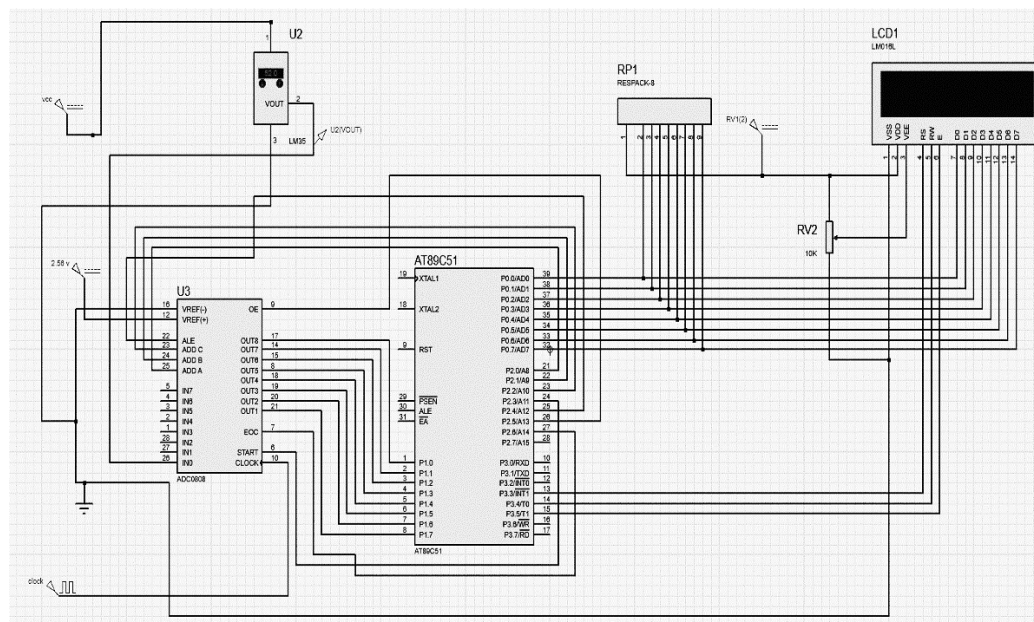


Fig 28.4 8051 connection to ADC and LM35

b) Simulation diagram

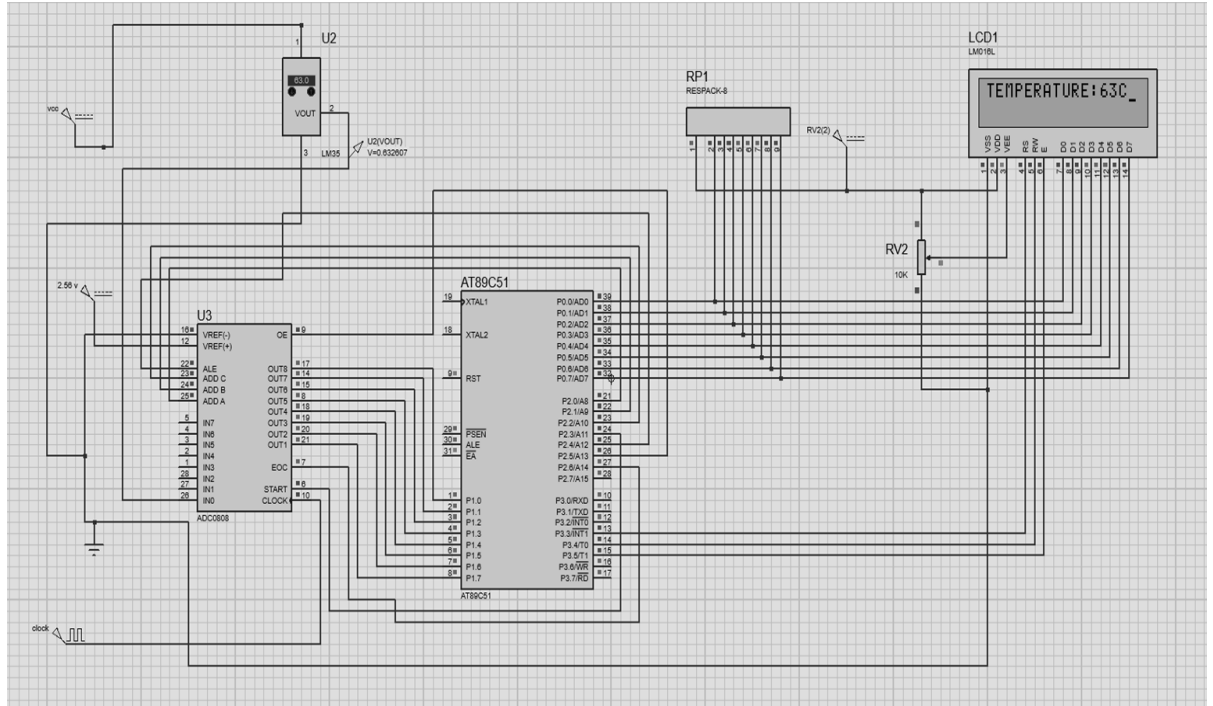


Fig 28.5 Simulation diagram

c) Actual circuit used in laboratory

VIII Required Resources/apparatus/equipment with specifications

Sr. No.	Instrument /Components	Specification	Quantity
1.	Microcontroller kit	Single board system with 8K RAM,ROM memory with battery backup,16X4,16X2LCD display, PC keyboard interfacing facility, Hex keypad facility, single user cross C-compiler,RS-232,USB, interfacing facility with built in power supply.	1 No.
2.	Desktop PC	Loaded with open source IDE, simulation and program downloading software.	1 No.
3.	ADC (0808) trainer board	Suitable to interface 8051 board.	1 No.
4.	LM 35 Temperature sensor	Suitable to interface ADC 0808	1 No.

IX Precautions to be followed

- 1) Care should be taken while operating LM 35 as it is rated to operate over -55°C to 150°C temperature range
- 2) Refer datasheet for to provide clock frequency to ADC 0808 chip.
- 3) Care must be taken while taking observations during power up.
- 4) Check rules / syntax of assembly language programming.

X Procedure

1. Write algorithm for given problem.
2. Draw flowchart for the same.
3. Develop assembly program using Integrated Development Environment (IDE) or any other relevant software tool.
4. Debug program on IDE.
5. Execute program on IDE.
6. Create hex file for the above program.
7. Download hex code in EPROM/Flash memory of the microcontroller.
8. Interface LM 35 to ADC 0808 IC and connect ADC output and LCD display to the microcontroller as per circuit diagram shown in fig 28.4
9. Vary temperature of LM 35 and observe ADC output on LCD display.

E-Waste Management

1. Identify pin configuration of the ICs and test the ICs on the IC tester.
2. If the IC is faulty then keep it in the proper e-waste bin.
3. If the IC is in OK condition, then mount it on breadboard or the trainer kit.
4. Utilize software-based simulations for training, decreasing the reliance on physical trainer kits and subsequently reducing e-waste

SAMPLE PROGRAM: Write a program to read the temperature, convert it to decimal and display on LCD

.Step 1-Algorithm

1. Define control signals RS, RW and EN for LCD and initialize LCD by sending commands.
2. Select an analog channel by providing bits to A, B, and C addresses according to the analog signal selection table.
3. Activate the ALE (address latch enable) pin.
4. Activate SC (start conversion) to initiate conversion.
5. Monitor EOC (end of conversion) to see whether conversion is finished. H-to-L output indicates that the data is converted and is ready to be picked up. If we do not use EOC, we can read the converted digital data after a brief time delay. The delay size depends on the speed of the external clock we connect to the CLK pin.
6. Activate OE (output enable) to read data out of the ADC chip.

Note: In ADC0808 that there is no self-clocking and the clock must be provided from an external source to the CLK pin. Although the speed of conversion depends on the frequency of the clock connected to the CLK pin, it cannot be faster than 100 microseconds.

Step 2-Flow Chart

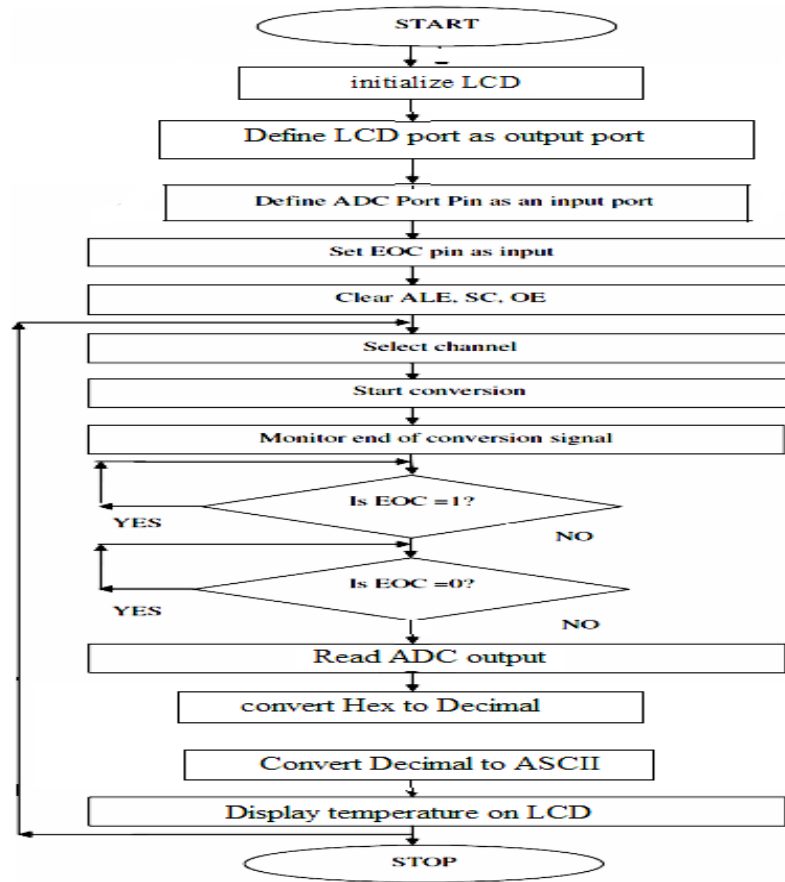


Fig 28.6 Flowchart for temperature measurement

Step 3-Assembly Language Program

Memory Address	Hex Code	Label	Mnemonics	Comments
			RS EQU P3.3	
			RW EQU P3.4	
			EN EQU P3.5	
			ALE BIT P2.0	
			SC BIT P2.1	
			EOC BIT P2.2	
			OE BIT P2.3	

Memory Address	Hex Code	Label	Mnemonics	Comments
			ADDR_A BIT P2.4	
			ADDR_B BIT P2.5	
			ADDR_C BIT P2.6	
			ORG 0000H	
C:0x0000	758000		MOV P0,#00H	;make P0 as output
C:0x0003	1163	BACK:	ACALL LCD	
C:0x0005	7590FF		MOV P1,#0FFH	;make P1 as input
C:0x0008	D2A6		SETB EOC	;make EOC an input
C:0x000A	C2A4		CLR ALE	;clear ALE
C:0x000C	C2A3		CLR SC	;clear WR
C:0x000E	C2A5		CLR OE	;clear RD
C:0x0010	C2A2		CLR ADDR_C	;C=0
C:0x0012	C2A1		CLR ADDR_B	;B=0
C:0x0014	C2A0		CLR ADDR_A	;A=0 (select channel 0)
C:0x0016	D2A4		SETB ALE	;latch address
C:0x0018	1131		ACALL DELAY	
C:0x001A	D2A3		SETB SC	;start conversion
C:0x001C	1131		ACALL DELAY	
C:0x001E	C2A4		CLR ALE	
C:0x0020	C2A3		CLR SC	
C:0x0022	20A6FD	HERE:	JB EOC,HERE	;wait
C:0x0025	30A6FD	HERE1:	JNB EOC,HERE1	
C:0x0028	D2A5		SETB OE	
C:0x002A	1131		ACALL DELAY	
C:0x002C	E590		MOV A,P1	;Read ADC data

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x002E	C2A5		CLR OE	
C:0x0030	1143		ACALL CONV	;call hex to ASCII
C:0x0032	1158		ACALL WRITETMP	; display temperature
C:0x0034	80CD		SJMP BACK	
C:0x0036	7B19	DELAY:	MOV R3,#25	;Delay Subroutine
C:0x0038	7C64	L3:	MOV R4,#100	
C:0x003A	7D64	L2:	MOV R5,#100	
C:0x003C	DDFE	L1:	DJNZ R5,L1	
C:0x003E	DCFA		DJNZ R4,L2	
C:0x0040	DBF6		DJNZ R3,L3	
C:0x0042	22		RET	
C:0x0043	75F00A	CONV:	MOV B,#10	; HEX to ASCII conversion
C:0x0046	84		DIV AB	
C:0x0047	AFF0		MOV R7,B	
C:0x0049	75F00A		MOV B,#10	
C:0x004C	84		DIV AB	
C:0x004D	AEF0		MOV R6,B	
C:0x004F	EE		MOV A,R6	
C:0x0050	2430		ADD A,#30H	
C:0x0052	FE		MOV R6,A	
C:0x0053	EF		MOV A,R7	
C:0x0054	2430		ADD A,#30H	
C:0x0056	FF		MOV R7,A	

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x0057	22		RET	
C:0x0058	EE		MOV A,R6	
C:0x0059	119E		ACALL LCDWRITE	
C:0x005B	EF		MOV A,R7	
C:0x005C	119E		ACALL LCDWRITE	
C:0x005E	7443		MOV A,#'C'	
C:0x0060	119E		ACALL LCDWRITE	
C:0x0062	22		RET	
C:0x0063	117C	LCD:	ACALL LCD_INIT	
C:0x0065	9000AB		MOV DPTR, #MSG	
C:0x0068	7A0C		MOV R2, #12	
C:0x006A	E4		UP:CLR A	
C:0x006B	93		MOVC A,@A+DPTR	
C:0x006C	119E		ACALL LCDWRITE	
C:0x006E	A3		INC DPTR	
C:0x006F	DAF9		DJNZ R2, UP	
C:0x0071	22		RET	
C:0x0072	7438	LCD_INIT:	MOV A, #38H	;LCD initialization
C:0x0074	1191		ACALL CMD	
C:0x0076	740E		MOV A, #0EH	
C:0x0078	1191		ACALL CMD	

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x007A	7401		MOV A, #01H	
C:0x007C	1191		ACALL CMD	
C:0x007E	7406		MOV A, #06H	
C:0x0080	1191		ACALL CMD	
C:0x0082	7480		MOV A, #80H	
C:0x0084	1191		ACALL CMD	
C:0x0086	22		RET	
C:0x0087	F580	CMD:	MOV P0, A	;LCD Command subroutine
C:0x0089	C2B3		CLR RS	
C:0x008B	C2B4		CLR RW	
C:0x008D	D2B5		SETB EN	
C:0x008F	C2B5		CLR EN	
C:0x0091	1136		ACALL DELAY	
C:0x0093	22		RET	
C:0x0094	F580	LCDWRITE:	MOV P0, A	;LCD data subroutine
C:0x0096	D2B3		SETB RS	
C:0x0098	C2B4		CLR RW	
C:0x009A	D2B5		SETB EN	
C:0x009C	C2B5		CLR EN	
C:0x009E	1136		ACALL DELAY	

Memory Address	Hex Code	Label	Mnemonics	Comments
C:0x00A0	22		RET	
		MSG:	DB "TEMPERATURE:"	
			END	

XI Resources Used

S. No.	Instrument /Components	Specification	Quantity
1.			
2.			
3.			

XII Actual Procedure Followed (use blank sheet provided if space not sufficient)

1.
2.
3.
4.
5.
6.
7.

XIII Observations for sample Program (use blank sheet provided if space not sufficient)

ADC input Voltage	Temperature displayed

XIV Results (Output of the Program)

.....
.....
.....

XV Interpretation of Results (Give meaning of the above obtained results)

.....
.....

XVI Conclusions and Recommendation (Actions/decisions to be taken based on the interpretation of results).

.....
.....

XVII Practical related questions

Note: Below given are few sample questions for reference. Teacher must design more such questions so as to ensure the achievement of identifies CO.

1. If output voltage of LM 35 sensor is 0.5V, then determine the Temperature detected.
2. If Vref pin is connected to 2.56V then calculate the step size of ADC0808.

[Space for Answers]

.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

XIX Assessment Scheme

Performance indicators		Weightage
Process related: 15 Marks		60%(15)
1	Use of IDE tools for programming	20%
2	Coding and Debugging ability	30%
3	Follow ethical practices.	10%
Product related: 10 Marks		40%(10)
4	Correctness of algorithm/ Flow chart	20%
5	Relevance of output of the problem definition	15%
6	Timely Submission of report, Answer to sample questions	05%
Total		100 %(25)

Marks Obtained			Dated signature of Teacher
Process Related (15)	Product Related (10)	Total (25)	