GOVERNMENT OF KARNATAKA
DEPARTMENT OF COLLEGIATE AND TECHNICAL EDUCATION

GOVERNMENT ENGINEERING COLLEGE

(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi)
Hyderabad Road, Yaramarus Camp,
RAICHUR – 584135
Ph. No:08532-200624,Fax:08532-251151,E-mail:ppl.gecr123@gmail.com

Department of Electronics & Communication Engineering

Microcontrollers Lab

(Ability Enhancement Course) IV

Semester B.E.

Sub Code: BECL456A

Academic Year: 2023-2024 EVEN SEM

Name : _____

USN : _____

Sem & Section : _____

GOVERNMENT OF KARNATAKA
DEPARTMENT OF COLLEGIATE AND TECHNICAL EDUCATION

GOVERNMENT ENGINEERING COLLEGE

(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi)
Hyderabad Road, Yaramarus Camp,
RAICHUR – 584135
Ph. No:08532-200624,Fax:08532-251151,E-mail:ppl.gecr123@gmail.com

Department of Electronics & Communication Engineering

Microcontrollers Lab

(Ability Enhancement Course)

Sub Code: BECL456A

Name : _____

USN : _____

Sem : _____

INSTITUTE VISION

To be center of Excellence of Higher Learning and Research in Engineering Imparting Ethical, Environmental and Economical aspects of the Society. .

INSTITUTE MISSION

Commitment in preparing Globally Competent Engineers in response to Rapid Technological growth through dynamic process of Teaching-Learning, Research and Professional Experiences for the Betterment of the Community.

.

DEPARTMENT VISION

Impart quality education to create world class technocrats and entrepreneurs with self-reliance to impart new ideas and innovations to excel in technical education and scientific research in Electronics & Communication Engineering.

DEPARTMENT MISSION

M1 : To Develop and deliver Quality academic programs in Emerging and innovative field of Engineering to empower the students to meet Industry Standards.

M2 : To provide domain affirmation, activities to add moral values and social awareness for the improvement in the effective engineering development, implementation and management of exploration based learning and to develop the thinking capability of young minds.

M3 : To create Centre of Excellence by establishing the incubation centers to meet global research challenges.

Program Outcomes (As Specified by NBA)

Engineering graduates will be able to:

PO1	**Engineering knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
PO2	**Problem analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences
PO3	**Design/development of solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations
PO4	**Conduct investigations of complex problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
PO5	**Modern tool usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
PO6	**The engineer and society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice
PO7	**Environment and sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development
PO8	**Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice
PO9	**Individual and team work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
PO10	**Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
PO11	**Project management and finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
PO12	**Life-long learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PEO (Program Educational Objectives)

PEO1: Excel in professional education in electronics & communication engineering with quality education.

PEO2: Exhibit professionalism, Ethics, communication skills, team work, self-dependence and right attitude in the profession as well adapt to the changes by continuous learning.

PEO3: Bring out the solution to real time problems in electronics & communication engineering developments.

PSO (Program Specific Outcomes)

PSO1: Specify, design, build and test analog and digital systems for signal processing including multimedia applications, using suitable components or simulation tools.

PSO2: Understand and architect wired and wireless analog and digital communication systems as per specifications, and determine their performance.

Do’s & Don’ts

Do’s inside the lab

- Each person may only use one computer at a time.
- All users should record the use of computer in the computer log book.
- Report any broken plugs or exposed electrical wires to the staff.
- Always save your progress.
- Remember to log out whenever you are done using any lab computer.

Don’ts inside the lab

- Computers and peripherals are not to be moved or reconfigured without approval of Lab In charge.
- Students should not install software on lab computers. If you have a question regarding specific software that you need to use, contact the staff.
- Food is not allowed in computer labs.
- Behavior and activities that disturb other users or disrupt the operations of the lab are not allowed.

Note: Internet facility is strictly for educational purposes.

COURSE OUTCOMES (With PO & PSO Mapping)

Course Outcomes		PO & PSO
CO1	Develop Assembly Language /C programs in 8051 for solving simple problems that manipulate input data using different instructions.	PO1, PO2, PO3, PO4, PO5, PO8, PO9, PO10 PSO1
CO2	Develop Testing and experimental procedures on 8051 Microcontroller, Analyze their operation under different cases.	PO1, PO2, PO3, PO4, PO5, PO8, PO9, PO10 PSO1
CO3	Develop programs for 8051 Microcontroller to implement real world problems.	PO1, PO2, PO3, PO4, PO5, PO8, PO9, PO10 PSO1
CO4	Develop Microcontroller applications using external hardware interface.	PO1, PO2, PO3, PO4, PO5, PO8, PO9, PO10, PO12, PSO1, PSO2

Correlation Matrix

CO/PO	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	P09	PO10	PO11	PO12	PSO1	PSO2	PSO2
CO1	2	2	2	3	2	-	-	2	2	3	-	-	2	-	-
CO2	2	2	1	3	2	-	-	2	3	3	-	-	2	-	-
CO3	2	3	3	3	2	-	-	2	3	3	-	-	2	-	-
CO4	3	3	3	3	2	-	-	2	3	3	-	1	2	1	1

Note: Correlation levels: 1: Slight (Low) 2: Moderate (Medium) 3: Substantial (High), "-" No correlation

Microcontrollers Lab Manual

Course Code	**BECL456A**	**CIE Marks**	**50**
Teaching Hours/Week (L:T:P)	**0:0:2**	**SEE Marks**	**50**
Credits	**1**	**Exam Hours**	**02**
Examination type (SEE)	**Practical**		

Course objectives: This course will enable students to:
- Understand the basic programming of Microcontrollers.
- Develop the 8051 Microcontroller-based programs for various applications using Assembly Language & C Programming.
- Program 8051 Microcontroller to control an external hardware using suitable I/O ports.

Sl. No.	**I. Assembly Language Programming**
Data Transfer Programs:	
1	Write an ALP to move a block of n bytes of data from source (20h) to destination (40h) using Internal-RAM.
2	Write an ALP to move a block of n bytes of data from source (2000h) to destination (2050h) using External RAM.
3	Write an ALP To exchange the source block starting with address 20h, (Internal RAM) containing N (05) bytes of data with destination block starting with address 40h (Internal RAM).
4	Write an ALP to exchange the source block starting with address 10h (Internal memory), containing n (06) bytes of data with destination block starting at location 00h (External memory).
Arithmetic & Logical Operation Programs:	
5	Write an ALP to add the byte in the RAM at 34h and 35h, store the result in the register R5 (LSB) and R6 (MSB), using Indirect Addressing Mode.
6	Write an ALP to subtract the bytes in Internal RAM 34h & 35h store the result in register R5 (LSB) & R6 (MSB).
7	Write an ALP to multiply two 8-bit numbers stored at 30h and 31h and store16- bit result in 32h and 33h of Internal RAM.
8	Write an ALP to perform division operation on 8-bit number by 8-bit number.
9	Write an ALP to separate positive and negative in a given array.
10	Write an ALP to separate even or odd elements in a given array.
11	Write an ALP to arrange the numbers in Ascending & Descending order.
12	Write an ALP to find Largest & Smallest number from a given array starting from 20h & store it in Internal Memory location 40h.

Counter Operation Programs:	
13	Write an ALP for Decimal UP-Counter.
14	Write an ALP for Decimal DOWN-Counter.
15	Write an ALP for Hexadecimal UP-Counter.
16	Write an ALP for Hexadecimal DOWN-Counter.
II. C Programming	
1	Write an 8051 C program to find the sum of first 10 Integer Numbers.
2	Write an 8051 C program to find Factorial of a given number.
3	Write an 8051 C program to find the Square of a number (1 to 10) using Look-Up Table.
4	Write an 8051 C program to count the number of Ones and Zeros in two consecutive memory locations.
III. Hardware Interfacing Programs	
1	Write an 8051 C Program to rotate stepper motor in Clock & Anti-Clockwise direction.
2	Write an 8051 C program to Generate Sine & Square waveforms using DAC interface.

Course Outcomes: At the end of the course the student will be able to:

1. Write an Assembly Language/ C programs in 8051for solving simple problems that manipulate input data using different instructions.
2. Develop testing and experimental procedures on 8051Microcontroller, analyze their operation under different cases.
3. Develop programs for 8051Microcontroller to implement real world problems.
4. Develop microcontroller applications using external hardware interface.

Assessment Details (both CIE and SEE)

The weightage of Continuous Internal Evaluation (CIE) is 50% and for Semester End Exam (SEE) is 50%. The minimum passing mark for the CIE is 40% of the maximum marks (20 marks out of 50) and for the SEE minimum passing mark is 35% of the maximum marks (18 out of 50 marks). A student shall be deemed to have satisfied the academic requirements and earned the credits allotted to each subject/ course if the student secures a minimum of 40% (40 marks out of 100) in the sum total of the CIE (Continuous Internal Evaluation) and SEE (Semester End Examination) taken together.

Continuous Internal Evaluation (CIE):

CIE marks for the practical course are **50 Marks**.

The split-up of CIE marks for record/ journal and test are in the ratio **60:40**.

- Each experiment is to be evaluated for conduction with an observation sheet and record write-up. Rubrics for the evaluation of the journal/write-up for hardware/software experiments are designed by the faculty who is handling the laboratory session and are made known to students at the beginning of the practical session.
- Record should contain all the specified experiments in the syllabus and each experiment write-up will be evaluated for 10 marks.
- Total marks scored by the students are scaled down to **30 marks** (60% of maximum marks).
- Weightage to be given for neatness and submission of record/write-up on time.

- Department shall conduct a test of 100 marks after the completion of all the experiments listed in the syllabus.
- In a test, test write-up, conduction of experiment, acceptable result, and procedural knowledge will carry a weightage of 60% and the rest 40% for viva-voce.
- The suitable rubrics can be designed to evaluate each student's performance and learning ability.
- The marks scored shall be scaled down to **20 marks** (40% of the maximum marks).

The Sum of scaled-down marks scored in the report write-up/journal and marks of a test is the total CIE marks scored by the student.

Semester End Evaluation (SEE):

- SEE marks for the practical course are 50 Marks.
- **SEE shall be conducted jointly by the two examiners of the same institute; examiners are appointed by the Head of the Institute**.
- The examination schedule and names of examiners are informed to the university before the conduction of the examination. These practical examinations are to be conducted between the schedule mentioned in the academic calendar of the University.
- All laboratory experiments are to be included for practical examination.
- (Rubrics) Breakup of marks and the instructions printed on the cover page of the answer script to be strictly adhered to by the examiners. **OR** based on the course requirement evaluation rubrics shall be decided jointly by examiners.
- Students can pick one question (experiment) from the questions lot prepared by the examiners jointly.
- Evaluation of test write-up/ conduction procedure and result/viva will be conducted jointly by examiners.

General rubrics suggested for SEE are mentioned here, writeup-20%, Conduction procedure and result in -60%, Viva-voce 20% of maximum marks. SEE for practical shall be evaluated for 100 marks and scored marks shall be scaled down to 50 marks (however, based on course type, rubrics shall be decided by the examiners) Change of experiment is allowed only once and 15% of Marks allotted to the procedure part are to be made zero.

The minimum duration of SEE is 02 hours.

Programming USING KIEL Simulator

Steps to **Create, Build, & Run** Project in Kiel:

1. Click the µVision symbol to open Kiel IDE.
2. In µV IDE Click the 'project' → 'New Project' option. 'create new project' window will be opened. Enter the 'file name' and click 'Save'.
3. Once 'select device for Taget 'Taget1'' window is opened. Under this double click 'Atmel' option select 'AT89C51' and click OK. Then click NO for popup message.
4. Click 'File' → 'New' file. Save the New file enter file name with "**.asm**" extension then click 'Save'.
5. If it is C-program Click 'File' → 'New' file. Save the New file enter file name with "**.C**" extension then click 'Save'
6. In 'Project Workspace' select 'Target1' and RIGHT click on 'Source Group1' and select 'Add Files to Group 'Source Group1''.
7. Select the 'files of Type' option for 'All files (*.*)' option. Enter the file name saved in the step 4. Click 'Add'. Observe that saved file name will be added in 'project workspace' under Target1 → Source Group1 → . then click 'close' option in popup message.
8. Then start writing ASM code in the file.
9. To **COMPILE** the code click 'Project' → 'Rebuild all target files'. In IDE 'BUILD' window check the message '"New" - 0 Error(s), 0 Warning(s) 'for compilation.
10. If Error is Zero indicate that the program has compiled without any error.
11. If Error are shown with numbers. Then go to particular line No. in the file and rectify the error and continue the compilation.
12. Once compilation done, To Start execution click 'Debug' → 'Start/Stop Debug Session'. Click 'OK'.
13. To **RUN** click 'Debug' → 'Run'. Check the results in the 'Register' 'Value'.

Introduction to Microcontroller 8051

The most universally employed set of microcontrollers come from the 8051 family. 8051 Microcontrollers persist to be an ideal choice for a huge group of hobbyists and experts. The original 8051 microcontroller was initially invented by Intel. The two other members of this 8051 family are-

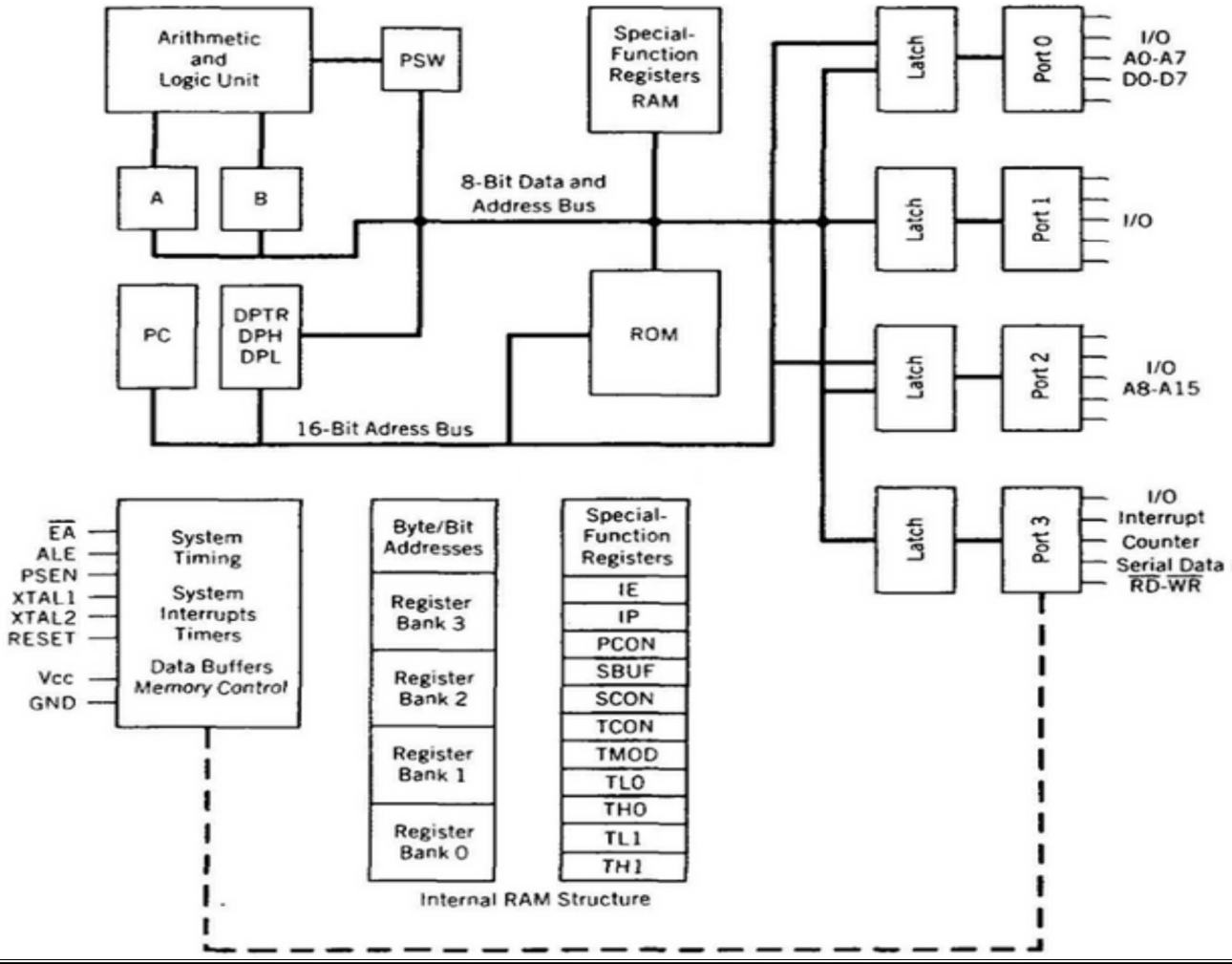
- 8052-This microcontroller has 3 timers & 256 bytes of RAM. Additionally it has all the features of the traditional 8051 microcontroller. 8051 microcontroller is a subset of 8052 microcontroller.
- 8031 - This microcontroller is ROM less, other than that it has all the features of a traditional 8051 microcontroller. For execution an external ROM of size 64K bytes can be added to its chip.

8051 microcontroller brings into 2 different sorts of memory such as - NV- RAM, UV - EPROM and Flash. 8051 is the basic microcontroller to learn embedded systems projects.

FEATURES OF 8051

1. 8051 microcontroller is an eight bit microcontroller.
2. It is available in 40 pin DIP package.
3. It has 4kb of ROM (on-chip programmable space) and 128 bytes of RAM space which is inbuilt, if desired 64KB of external memory can be interfaced with the microcontroller.
4. There are four parallel 8 bits ports which are easily programmable as well as addressable.
5. An on- chip crystal oscillator is integrated in the microcontroller which has crystal frequency of 12MHz.
6. In the microcontroller there is a serial input/output port which has 2 pins.
7. Two timers of 16 bits are also incorporated in it; these timers can be employed as timer for internal functioning as well as counter for external functioning.
8. The microcontroller comprise of 5 interrupt sources namely- Serial Port Interrupt, Timer Interrupt 1, External Interrupt 0, Timer Interrupt 0, External Interrupt 1.
9. The programming mode of this micro-controller includes GPRs (general purpose registers), SFRs (special function registers) and SPRs (special purpose registers)

INTERNAL ARCHITECHURE OF 8051 MICRO-CONTROLLER

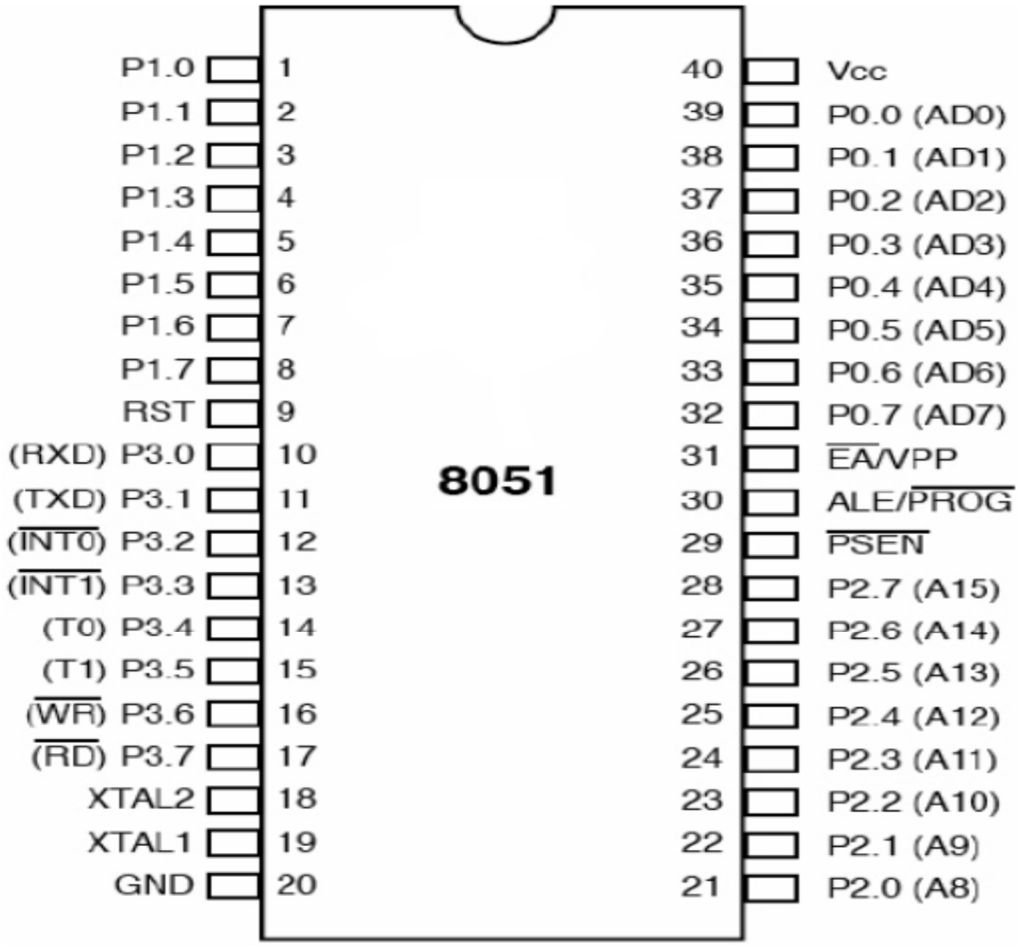


1**. ALU**: All arithmetic and logical functions are carried out by the ALU. Addition, subtraction with carry, and multiplication come under arithmetic operations. Logical AND, OR and exclusive OR (XOR) come under logical operations.

2. **Program Counter (PC)** : A program counter is a 16-bit register and it has no internal address. The basic function of program counter is to fetch from memory the address of the next instruction to be executed. The PC holds the address of the next instruction residing in memory and when a command is encountered, it produces that instruction. This way the PC increments automatically, holding the address of the next instruction.

3. **Registers** : Registers are usually known as data storage devices. 8051 microcontroller has 2 registers, namely Register A and Register B. Register A serves as an accumulator while Register B functions as a general purpose register. These registers are used to store the output of mathematical and logical instructions. The operations of addition, subtraction, multiplication and division are carried out by Register A. Register B is usually unused and comes into picture only when multiplication and division functions are carried out by Register A. Register A also involved in data transfers between the microcontroller and external memory.

PIN DIAGRAM OF 8051 MICRO-CONTROLLER



PINOUT DESCRIPTION

Pins 1-8	**Port 1** Each of these pins can be configured as an input or an output.
Pin 9	**RS** A logic one on this pin disables the microcontroller and clears the contents of most registers. In other words, the positive voltage on this pin resets the microcontroller. By applying logic zero to this pin, the program starts execution from the beginning
Pins10-17	**Port 3** Similar to port 1, each of these pins can serve as general input or output. Besides, all of them have alternative functions
Pin 10	**RXD** Serial asynchronous communication input or Serial synchronous communication output
Pin 11	**TXD** Serial asynchronous communication output or Serial synchronous communication clock output
Pin 12	**INT0** Interrupt 0 input
Pin 13	**INT1** Interrupt 1 input
Pin 14	**T0** Counter 0 clock input
Pin 15	**T1** Counter 1 clock input
Pin 16	**WR** Write to external (additional) RAM
Pin 17	**RD** Read from external RAM
Pin 18, 19	**$XTAL_2$, $XTAL_1$** are internal oscillator input and output pins. A quartz crystal which specifies operating frequency is usually connected to these pins. Instead of it, miniature ceramics resonators can also
Pin 20	**GND** Ground
Pin 21-28	**Port 2** If there is no intention to use external memory then these port pins are configured as general inputs/outputs. In case external memory is used, the higher address byte, i.e. addresses A8-A15 will appear on this port. Even though memory with capacity of 64Kb is not used, which means that not all eight port bits are used for its addressing, the rest of them are not available as inputs/outputs
Pin 29	**PSEN** If external ROM is used for storing program then a logic zero (0) appears on it every time the microcontroller reads a byte from memory
Pin 30	**ALE** Prior to reading from external memory, the microcontroller puts the lower address byte (A0-A7) on P0 and activates the ALE output. After receiving signal from the ALE pin, the external register (usually 74HCT373 or 74HCT375 add-on chip) memorizes the state of P0 and uses it as a memory chip address. Immediately after that, the ALU pin is returned its previous logic state and P0 is now used as a Data Bus. As seen, port data multiplexing is performed by means of only one additional (and cheap) integrated circuit. In other words, this port is used for both data and address transmission
Pin 31	**EA** By applying logic zero to this pin, P2 and P3 are used for data and address transmission with no regard to whether there is internal memory or not. It means that even there is a program written to the microcontroller, it will not be executed. Instead, the program written to external ROM will be executed. By applying logic one to the EA pin, the microcontroller will use both memories, first internal then external (if exists).
Pin 32-39	**Port 0** Similar to P2, if external memory is not used, these pins can be used as general inputs/outputs. Otherwise, P0 is configured as address output (A0-A7) when the ALE pin is driven high (1) or as data output (Data Bus) when the ALE pin is driven low (0).
Pin 40	**VCC** +5V power supply

.

Contents

I. Assembly Programming

Data Transfer Programs

1. Write an ALP to move a block of n bytes of data from source (20h) to destination (40h) using Internal RAM.

```
ORG 0000H
MOV   R0,   #20H
MOV   R1,   #40H
MOV R2, #05H
back : MOV A , @R0
MOV @R1, A
INC R0
INC R1
DJNZ  R2, back
exit:  SJMP exit
end
```

Before Execution:

Input@ D:20h	01	02	03	04	05
Output@ D:40h					

After Execution:

Input@ D:20h	01	02	03	04	05
Output@ D:40h	01	02	03	04	05

2. Write an ALP to move a block of n bytes of data from source (2000h) to destination (2050h) using External RAM.

```
ORG 0000H
MOV DPTR, #2000H
MOV R0, #00H
MOV R1, #50H
MOV A, #00H
MOV R7,#07H
BACK: MOVX A,@DPTR
MOV DPL, R1
MOVX @DPTR, A
INC R1
INC R0
MOV DPL, R0
DJNZ R7,BACK
exit:  SJMP exit
END
```

Before Execution:

Input@ X:2000h	01	02	03	04	05	06	07
Output@ X:2050h							

After Execution:

Input@ X:2000h	01	02	03	04	05	06	07
Output@ X:2050h	01	02	03	04	05	06	07

3. Write an ALP To exchange the source block starting with address 20h, (Internal RAM) containing N (05) bytes of data with destination block starting with address 40h (Internal RAM).

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

Before Execution:

Input@ D:20h	01	02	03	04	05
Input@ D:40h	0A	0B	0C	0D	0E

After Execution:

output@ D:20h	0A	0B	0C	0D	0E
Ouput@ D:40h	01	02	03	04	05

4. Write an ALP to exchange the source block starting with address 10h (Internal memory), containing n (06) bytes of data with destination block starting at location 00h (External memory).

```
        ORG 0000H
        MOV R1, #06
        MOV R0, #10H
        MOV DPTR, #0000H
back:   MOVX A, @DPTR
        XCH A, @R0
        MOVX @DPTR, A
        INC R0
        INC DPTR
        DJNZ R1, back
exit:   SJMP exit

        END
```

Before Execution:

Input@ X:0000h	0A	0B	0C	0D	0E	0F
Intput@ D:10h	01	02	03	04	05	06

After Execution:

Output@ X:0000h	01	02	03	04	05	06
Output@ D:10h	0A	0B	0C	0D	0E	0F

Arithmetic & Logical Operation Programs

5. Write an ALP to add the byte in the RAM at 34h and 35h, store the result in the register R5 (LSB) and R6 (MSB), using Indirect Addressing Mode.

```
ORG 0000H
MOV R0, #34H
MOV R1, #35H
MOV R6, #00H
MOV A, @R0
BACK: ADD A, @R1
JNC NOCARY
INC R6
NOCARY: MOV R5, A
exit:  SJMP exit
END
```

Before Execution:

I/P		O/P	
34H	35H	R5	R6
FF	FF		

After Execution:

I/P		O/P	
34H	35H	R5	R6
FF	FF	FE	01

6. Write an ALP to subtract the bytes in Internal RAM 34h & 35h store the result in register R5 (LSB) & R6 (MSB).

```
ORG 8000H
MOV R0, #34H
MOV R1, #35H
CLR C
MOV A, @R0
MOV B, @R1
SUBB A, B
MOV R5, A
MOVC R6,
HERE:JMP HERE
END
```

Before Execution:

I/P		O/P	
30H	31H	R5	R6
05	04		

After Execution:

I/P		O/P	
30H	31H	R5	R6
05	04	02	

7. Write an ALP to multiply two 8-bit numbers stored at 30h and 31h and store16-bit result in 32h and 33h of Internal RAM.

```
ORG 0000H
MOV A, 30H
MOV B,31H
MUL AB
MOV 32H, A
MOV 33H,B
LOOP: SJMP LOOP
END
```

Before Execution:

I/P		O/P	
30H	31H	32H	33H
05	04		

After Execution:

I/P		O/P	
30H	31H	32H	33H
08	02	14	

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

```
ORG 0000H
MOV A, 40H
MOV  B,41H
DIV AB

MOV 42H, A
MOV 43H, B
LOOP: SJMP
LOOP

END
```

Before Execution:

I/P		O/P	
40H	41H	42H	43H
08	02		

After Execution:

I/P		O/P	
40H	41H	42H	43H
08	02	04	

9. Write an ALP to separate positive and negative in a given array.

```
ORG      0000H
MOV R2, #0AH
MOV R0, #20H
MOV R1, #40H
MOV DPTR, #2000H
CLR C
BACK: MOVX A, @DPTR
MOV R5, A
RLC A
JC NEG
MOVX A, @DPTR
MOV @R0, A
INC R0
SJMP NEXT
NEG: MOVX A, @DPTR
MOV @R1,A
INC R1
NEXT: INC DPTR
DJNZ R2, BACK
EXIT:SJMP EXIT
END
```

Before Execution:

Input@ X:2000h	01	05	AB	09	1B	CD	2D	FE	78	16
Output@ D:20h										
D:40h										

After Execution:

Input@ X:2000h	01	05	AB	09	1B	CD	2D	FE	78	16
Output@ D:20h	01	05	09	1B	2D	78	16			
D:40h	AB	CD	FE							

10. Write an ALP to separate even or odd elements in a given array.

```
ORG      0000H
MOV R2, #0AH
MOV R0, #20H
MOV R1, #40H
MOV DPTR, #2000H
CLR C
BACK: MOVX A, @DPTR
MOV R5, A
RRC A
JC ODD
MOVX A, @DPTR
MOV @R0, A
INC R0
SJMP NEXT
ODD: MOVX A, @DPTR
MOV @R1,A
INC R1
NEXT: INC DPTR
DJNZ R2, BACK
EXIT:SJMP EXIT
END
```

Before Execution:

Input@ X:2000h	05	1A	20	13	54	66	33	44	09	BC
Output@ D:20h										
D:40h										

After Execution:

Input@ X:2000h	05	1A	20	13	54	66	33	44	09	BC
Output@ D:20h	1A	20	54	66	44	BC				
D:40h	05	13	33	09						

11. Write an ALP to arrange the numbers in Ascending & Descending order.

Ascending order:

```
        ORG 0000H
        MOV R0, #09H
AGAIN:  MOV DPTR, #9000H
        MOV R1, #09H
BACK:   MOV R2, DPL
        MOVX A, @DPTR
        MOV B, A
        INC DPTR
        MOVX A, @DPTR
        CJNE A, B, LOOP
        AJMP SKIP
LOOP:   JNC SKIP
        MOV DPL, R2
        MOVX @DPTR, A
        INC DPTR
        MOV A, B
        MOVX @DPTR, A
SKIP:   DJNZ R1, BACK
        DJNZ R0, AGAIN
HERE:   JMP HERE
        END
```

INPUT		OUTPUT
X:9000 :	12	01
9001 :	19	02
9002 :	01	03
9003 :	02	09
9004 :	09	12
9005 :	72	16
9006 :	68	19
9007 :	16	68
9008 :	03	72

Descending order:

```
        ORG 0000H
        MOV R0, #09H
AGAIN:  MOV DPTR, #9000H
        MOV R1, #09H
BACK:   MOV R2, DPL
        MOVX A, @DPTR
        MOV B, A
        INC DPTR
        MOVX A, @DPTR
        CJNE A, B, LOOP
        AJMP SKIP
LOOP:   JC SKIP
        MOV DPL, R2
        MOVX @DPTR, A
        INC DPTR
        MOV A, B
        MOVX @DPTR, A
SKIP:   DJNZ R1, BACK
        DJNZ R0, AGAIN
HERE:   JMP HERE
        END
```

INPUT		OUTPUT
X:9000 :	32	93
9001 :	41	92
9002 :	45	59
9003 :	46	46
9004 :	59	45
9005 :	32	41
9006 :	93	32
9007 :	92	32
9008 :	26	26

12. Write an ALP to find Largest & Smallest number from a given array starting from 20h & store it in Internal Memory location 40h.

Largest Number in an array

```
        ORG 0000H
        MOV R2, #04H
        MOV A, #00H
        MOV R0, #20H
        MOV R1, #40H
UP:     MOV B, @R0
        CJNE A, B, DOWN
DOWN:   JNC BELOW
        MOV A, @R0
BELOW:  INC R0
        DJNZ R2, UP
        MOV @R1, A
HERE:   JMP HERE
        END
```

INPUT		OUTPUT	
D:20 :	FA	D:40:	FF
21 :	01		
22 :	FF		
23 :	60		

Smallest Number in an array

```
        ORG 0000H
        MOV R2, #04H
        MOV A, #00H
        MOV R0, #20H
        MOV R1, #40H
UP:     MOV B, @R0
        CJNE A, B, DOWN
DOWN:   JC BELOW
        MOV A, @R0
BELOW:  INC R0
        DJNZ R2, UP
        MOV @R1, A
HERE:   JMP HERE
        END
```

INPUT		OUTPUT	
D:20 :	FA	D:40:	01
21 :	01		
22 :	FF		
23 :	60		

Counter Operation Programs

13. Write an ALP for Decimal UP-Counter.

```
ORG 0000H
MOV R2, #00H
BACK:INC R2
CJNE R2, #0AH, LOOP
MOV R2, #00H
LOOP: SJMP BACK
END
```

Output:

OUTPUT (R2)
00
01
02
...
0A

14. Write an ALP for Decimal DOWN-Counter.

```
        ORG 0000H
        MOV R2, #09H
BACK:   DEC R2
CJNE R2, #00H, LOOP
        MOV R2, #09H
LOOP:   SJMP BACK
        END
```

Output:

OUTPUT (R2)
09
08
07
...
01
00

15. Write an ALP for Hexadecimal UP-Counter.

```
        ORG 0000H
        MOV R2, #00H
        BACK:    INC R2
        CJNE R2, #0FH, LOOP
       MOV R2, #00H
        LOOP:   SJMP BACK
                END
```

Output:

OUTPUT (R2)
00
01
02
…
0F

16. Write an ALP for Hexadecimal DOWN-Counter.

```
        ORG 0000H
        MOV R2, #0FH
BACK:   DEC R2
        CJNE R2, #00H, LOOP
        MOV R2, #0FH
LOOP:   SJMP BACK
        END
```

Output:

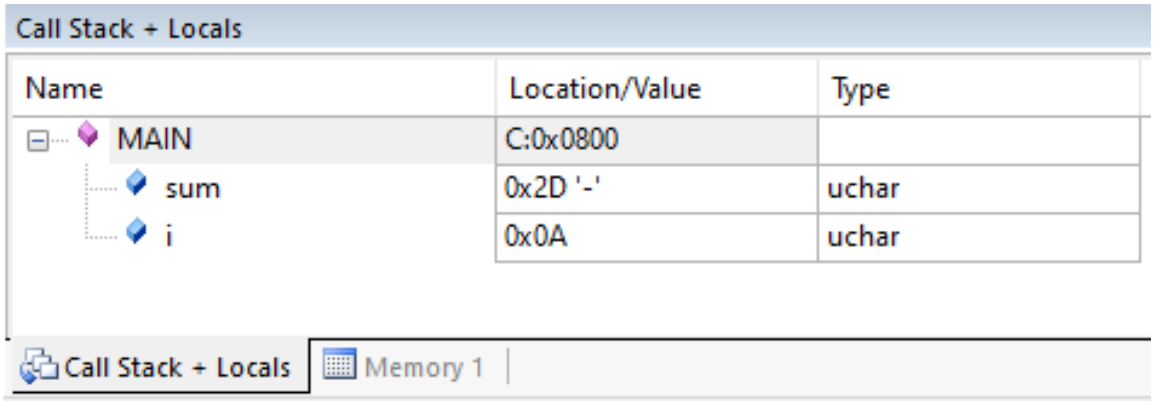
OUTPUT (R2)
0F
0E
0D
--
02
01
00

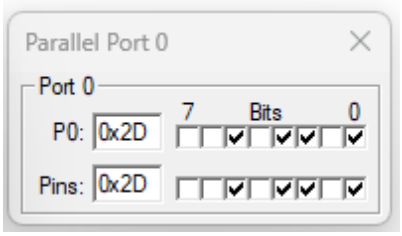
II. C Programming

1. Write a 8051 C program to find the sum of first 10 integer numbers.

```
#include <reg51.h>
void main ( )
        {
                unsigned char sum=0;
                unsigned char i;
                for (i=0; i<=9; i++)
                {
                        sum = sum + i;
                }
                ACC=sum;
                P0=sum;
        }
```

Output:

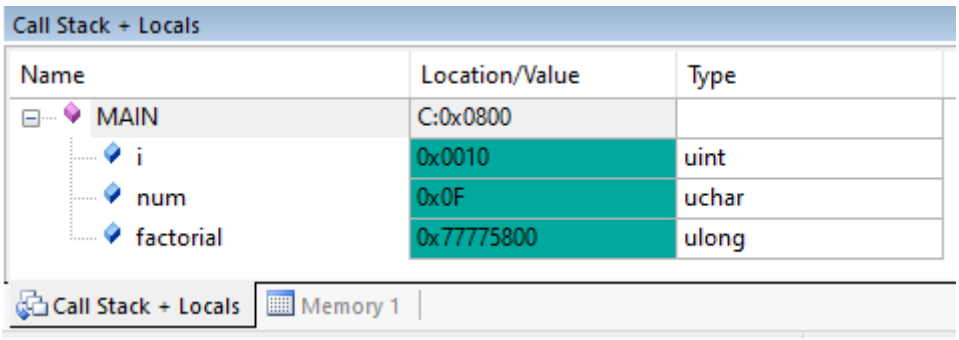


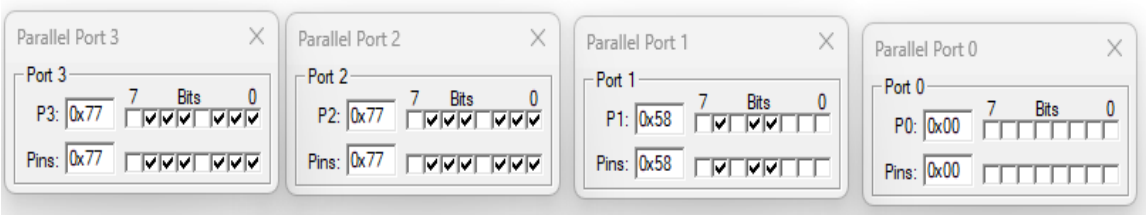


2. Write a 8051 C program to find factorial of a given number.

```
#include <reg51.h>
#include <stdio.h>
void main()
	{
		unsigned int i;
		unsigned char num = 15; // The number to find the factorial of
		unsigned long factorial = 1;
		for ( i = 1; i <= num; i++)
			{
				factorial = factorial*i;
			}
				P0= factorial;
				P1= (factorial & 0xff00) >>8;
				P2= (factorial & 0xff0000) >>16;
				P3= (factorial & 0xff000000) >>24;
	}
```

Output:





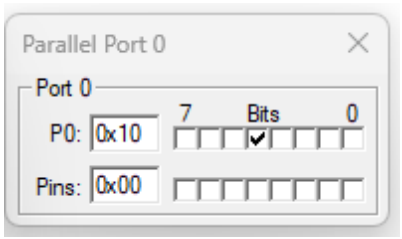
3. Write a 8051 C program to find the square of a number (1 to 10) using look-up table.

```
#include <reg51.h>
void main ( )
        {
                unsigned char LUT[]={1,4,9,16,25,36,49,64,81,100};
                unsigned char num, square;
                for(num=1; num<=10; num++)
                        {
                                square =LUT[num-1];
                                P0=square;
                        }
        }
```

Output:

Call Stack + Locals

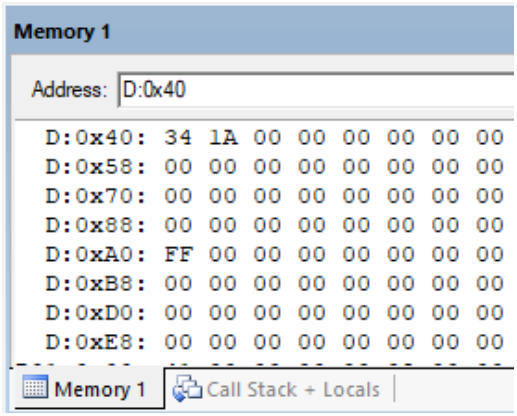
Name	Location/Value	Type
MAIN	C:0x08F6	
LUT	D:0x08 "□□\t□□$1@...	array[10] of uchar
[0]	0x01	uchar
[1]	0x04	uchar
[2]	0x09	uchar
[3]	0x10	uchar
[4]	0x19	uchar
[5]	0x24 '$'	uchar
[6]	0x31 '1'	uchar
[7]	0x40 '@'	uchar
[8]	0x51 'Q'	uchar
[9]	0x64 'd'	uchar
num	0x05	uchar
square	0x19	uchar

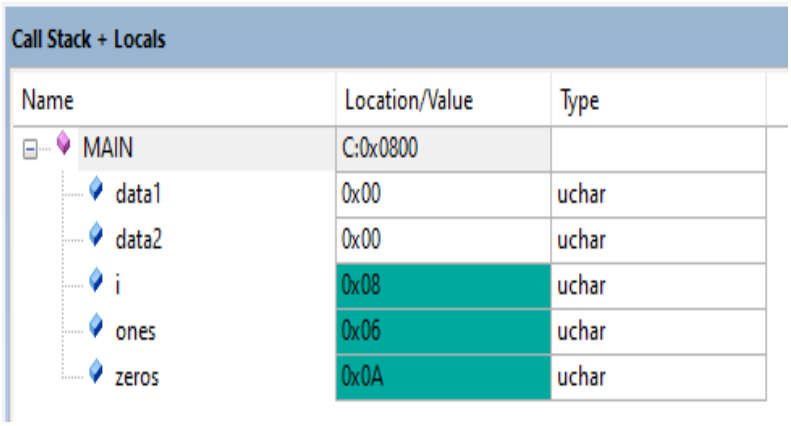


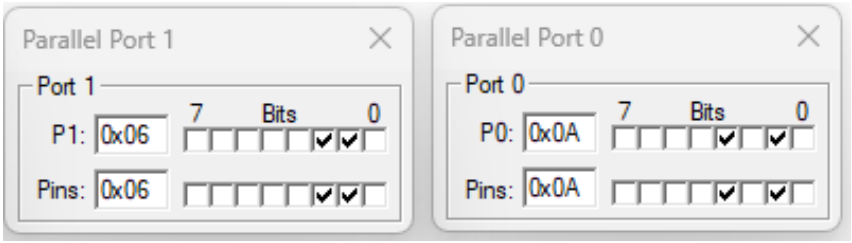
4. Write a 8051 C program to count the number of ones and zeros in two consecutive memory locations.

```
#include <reg51.h>
void main ( )
{
unsigned char data1, data2;
unsigned char i,ones, zeros;
data1 = *(unsigned char*) 0x40;
data2 = *(unsigned char*) 0x41;
CY=0;
for(i=0;i<=7;i++)
{
data1 >>=1;
if(CY==1) ones++;
else zeros++;
}
for(i=0;i<=7;i++)
{
data2 >>=1;
if(CY==1) ones++;
else zeros++;
}
P0=zeros;
P1=ones;
}
```

Output:





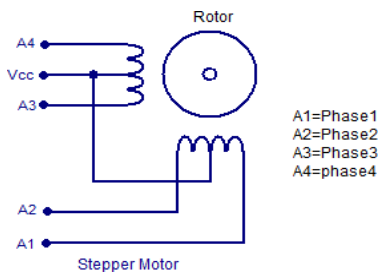


III. Hardware Interfacing Programs

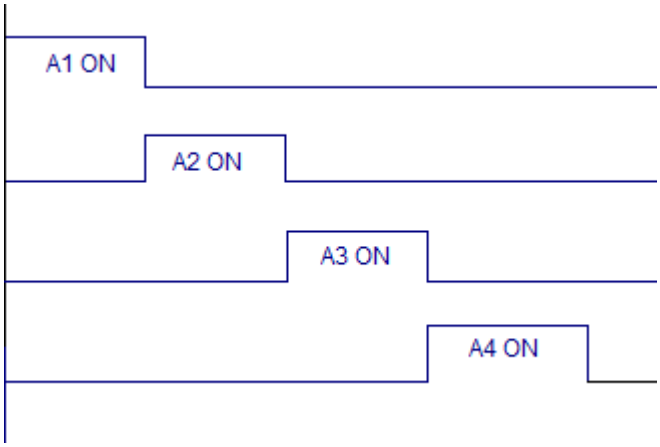
1. Write an 8051 C Program to rotate stepper motor in Clock & Anti-Clockwise direction.

```
#include<reg51.h>
void ms_delay(unsigned int t) //To create a delay of 200 ms = 200 x 1ms
   {
   unsigned i,j ;
   for(i=0;i<t;i++)      //200 times 1 ms delay
   for(j=0;j<1275;j++);  //1ms delay
   }
void main()
{
   while(1) // To repeat infinitely
   {
      P2=0x0C;         //P2 = 0000 1000 First Step
      ms_delay(100);
      P2=0x06;         //P2 = 0000 0100 Second Step
      ms_delay(100);
      P2=0x03;         //P2 = 0000 0010 Third Step
      ms_delay(100);
      P2=0x09;         //P2 = 0000 0001 Fourth Step
      ms_delay(100);

                  P2=0x09;         //P2 = 0000 1000 First Step
      ms_delay(100);
      P2=0x03;         //P2 = 0000 0100 Second Step
      ms_delay(100);
      P2=0x06;         //P2 = 0000 0010 Third Step
      ms_delay(100);
      P2=0x0C;         //P2 = 0000 0001 Fourth Step
      ms_delay(100);

                  }
}
```

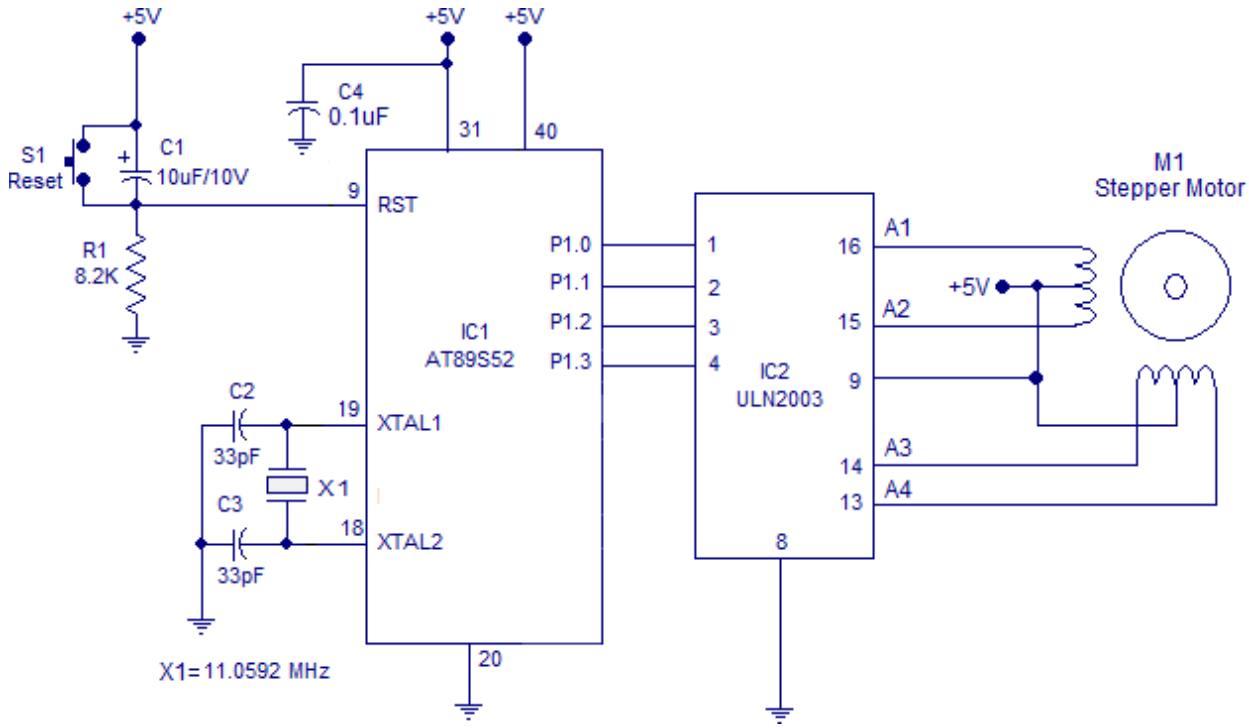
Stepper Motor:



The stepper motor is rotated by switching individual phases ON for a given time one by one. The sequence is given in the graph below.



Circuit diagram.

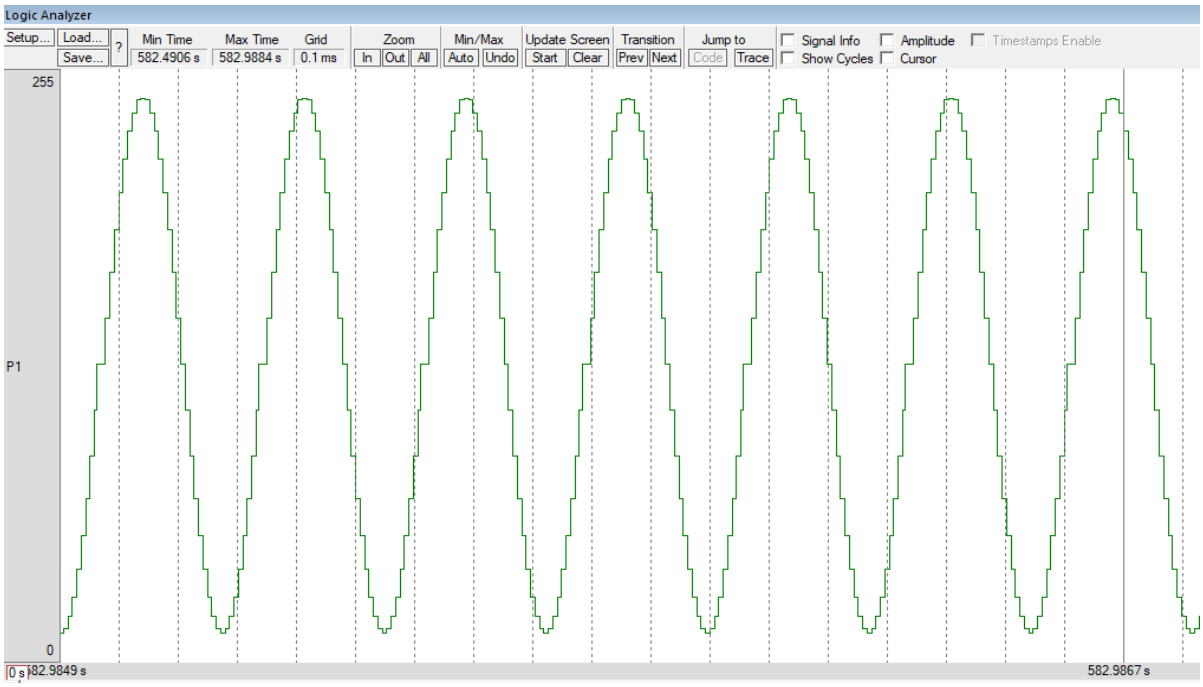


2. Write an 8051 C program to generate sine and square waveforms using DAC interface.

Sine Wave:

```
#include<reg51.h>
int main()
{
int j;
int
c[37]={128,150,172,192,210,226,239,248,254,255,254,248,239,226,210,192,172,150,1
28,106,84,64,46,30,17,8,2,0,2,8,17,30,46,64,84,106,128};
while(1)
{
for(j=0;j<36;j++)
{
P1=c[j];
}
P1=128;
}}
```

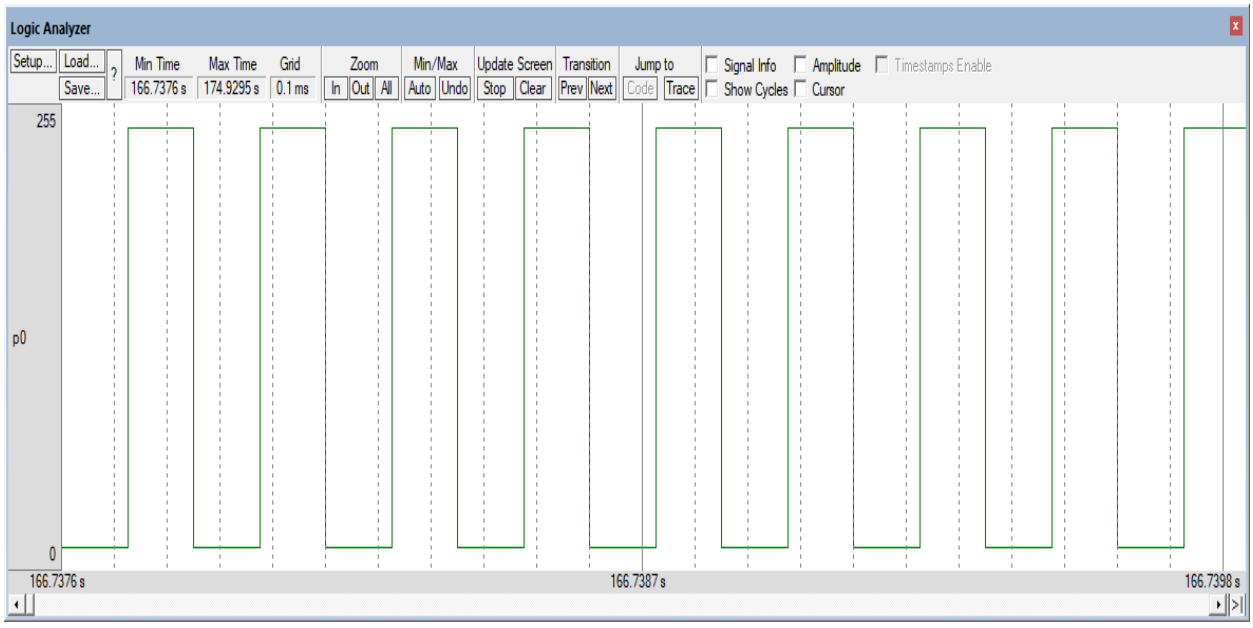
Output:



Square Wave:

```
#include <reg51.h>
void delay()
{
int i=0;
for(i=0;i<15;i++)
        {
}
        }
void main()
{
        while(1)
        {
P0=0xff;
delay();
P0=0x00;
delay();
}
}
```

Output:



Viva Questions

1. Intel 8051 follows which architecture?
   Intel 8051 is Harvard Architecture.
2. What is the difference between Harvard Architecture and von Neumann Architecture?
   The name Harvard Architecture comes from the Harvard Mark. The most obvious characteristic of the Harvard Architecture is that it has physically separate signals and storage for code and data memory. It is possible to access program memory and data memory simultaneously. Typically, code (or program) memory is read-only and data memory is read-write. Therefore, it is impossible for program contents to be modified by the program itself.

   The von Neumann architecture is named after the mathematician and early computer scientist John von Neumann. Von Neumann machines have shared signals and memory for code and data. Thus, the program can be easily modified by itself since it is stored in read-write memory.
3. 8051 was developed using which technology?
   Intel's original MCS-51 family was developed using NMOS technology, but later versions, identified by a letter C in their name (e.g., 80C51) used CMOS technology and consume less power than their NMOS predecessors. This made them more suitable for battery-powered devices.
4. Why 8051 is called an 8-bit microcontroller?
   The Intel 8051 is an 8-bit microcontroller which means that most available operations are limited to 8 bits.
5. What is the width of the data bus?
   8-bit data bus
6. What is the width of the address bus?
   16-bit address bus (PC -16 bit wide).
7. List the features of the 8051 microcontrollers?

   40 Pin IC.
   128 bytes of RAM.
   4K ROM (On-chip and could be different for different versions).
   2 Timers (Timer 0 and Timer 1).
   32 Input/ Output pins.
   1 serial port.
   6 Interrupts (Including Reset).
8. What location code memory space and data memory space begins?
   At location 0x00 for internal or external memory
9. What is 8051 Microcontroller ?

   The intel 8051 microcontroller is one of the most popular general-purpose microcontrollers in use today. It is an 8-bit family of microcontroller developed by Intel in the year 1981. This microcontroller was also referred to as "system on a chip" because it has 128 bytes of RAM, 4Kbytes of ROM, 2 Timers, 1 Serial port,

and four ports on a single chip. 8051 microcontroller allows CPU to work on 8bits of data at a time. In case the data is larger than 8 bits then it has to be broken into parts so that the CPU can process conveniently.

10. What are registers in Microcontroller ?

    Register provides a fast way to collect and store data using microcontrollers and processors. If we want to manipulate data with a controller or processor by performing tasks like addition, subtraction, and so on, we cannot do that directly in the memory, in order to perform these tasks we need registers to process and store the data. Microcontrollers contain several types of registers that can be classified according to their content or instructions that operate on them.

    The 8051 microcontroller contains mainly two types of registers:

    - General purpose registers (Byte addressable registers)
    - Special function registers (Bit addressable registers)

11. Which are the most common 8051 series of microcontrollers?

    Atmel series AT89C2051 and Philips family P89C51RD2 are the two most common microcontrollers of 8051 families.

12. What is the internal RAM size of 8051

    128 bytes

13. When 8051 wakes up then 0x00 is loaded to which register?

    Program Counter

14. How many bytes of bit addressable memory is present in 8051 based microcontrollers?

    16 bytes

15. List Interrupts available in 8051 microcontrollers.

    - External interrupt 0 (IE0) has highest priority among interrupts.
    - Timer interrupt 0 (TF0)
    - External interrupt 1 (IE1)
    - Timer interrupt 1 (TF1) has lowest priority among other interrupts.
    - Serial port Interrupt
    - Reset

16. What is the meaning of the instruction MOV A,05H?

    Address 05H is stored in the accumulator.

17. How are the status of the carry, auxiliary carry and parity flag affected if the write instruction

    MOVA,#9C

    ADD A,#64H

    CY=1,AC=1,P=0
18. When the microcontroller executes some arithmetic operations, then the flag bits of which register are affected?

    PSW
19. How are the bits of the register PSW affected if we select Bank2 of 8051?

    PSW.3=0 and PSW.4=1
20. DJNZ R0, label is how many bit instructions?

    2
21. JZ, JNZ, DJNZ, JC, JNC instructions monitor the bits of which register?

    PSW
22. DAA command adds 6 to the nibble if:

    Either CY or AC is 1
23. What is the clock source for the timers?

    From the crystal applied to the micro-controller
24. What is the function of the TMOD register?

    TMOD register is used to set different timer's or counter's to their appropriate modes
25. Auto reload mode is allowed in which mode of the timer?

    Mode 2
26. A counter is fundamentally a _____sequential circuit that proceeds through the predetermined sequence of states only when input pulses are applied to it. Register
27. Which special function register play a vital role in the timer/counter mode selection process by allocating the bits in it?

    TMOD
28. ANL instruction is used _____

    To AND the contents of the two registers and to mask the status of the bits
29. CJNE instruction makes _____

In CJNE command, the pointer jumps if the values of the two registers are not equal and it resets CY if the destination address is larger then the source address and sets CY if the destination address is smaller then the source address.

30. XRL, ORL, ANL commands have accumulator as the destination address and any register, memory or any immediate data as the source address
31. In unsigned number addition, the status of which bit is important? CY
32. Which of these instructions have no effect on the flags of PSW?
    ANL, ORL and XRL.These instructions are the arithmetic operations and the flags are affected by the data copy instructions, so all these instructions don't affect the bits of the flag.
33. What does ASCII stand for?
    American Standard Code for Information Interchange. The ASCII codes are used to represent the bits into symbols and vice versa. ASCII is the American Standard Code which is used to exchange information.
34. Binary Coding for the letter X is _____ 01011000
35. Binary coded decimal is a combination of _____
    Four binary digits. Binary coded decimal is a combination of 4 binary digits. For example-8421.
36. Add the two BCD numbers: 1001 + 0100 = ?
    00010011. Firstly, Add the 1001 and 0100. We get 1101 as output but it's not in BCD form. So, we add 0110 (i.e. 6) with 1101. As a result we get 10011 and it's BCD form is 0001 0011.
37. Which is the number system that uses numbers as well as alphabets? Hexadecimal System

Data transfer instructions.

In this group, the instructions perform data transfer operations of the following types.

a. Move the contents of a register Rn to A

i. MOV A,R2

ii. MOV A,R7

b. Move the contents of a register A to Rn

i. MOV R4,A

ii. MOV R1,A

c. Move an immediate 8 bit data to register A or to Rn or to a memory location(direct or indirect)

i. MOV A, #45H

ii. MOV R6, #51H

iii. MOV 30H, #44H

iv. MOV @R0, #0E8H

v. MOV DPTR, #0F5A2H

vi. MOV DPTR, #5467H

d. Move the contents of a memory location to A or A to a memory location using direct and indirect addressing

i. MOV A, 65H

ii. MOV A, @R0

iii. MOV 45H, A

iv. MOV @R1, A

e. Move the contents of a memory location to Rn or Rn to a memory location using direct addressing

i. MOV R3, 65H

ii. MOV 45H, R2

f. Move the contents of memory location to another memory location using direct and indirect addressing

i. MOV 47H, 65H

ii. MOV 45H, @R0

g. Move the contents of an external memory to A or A to an external memory

i. MOVX A,@R1

ii. MOVX @R0,A

iii. MOVX A,@DPTR

iv. MOVX@DPTR,A

h. Move the contents of program memory to A

i. MOVC A, @A+PC

ii. MOVC A, @A+DPTR

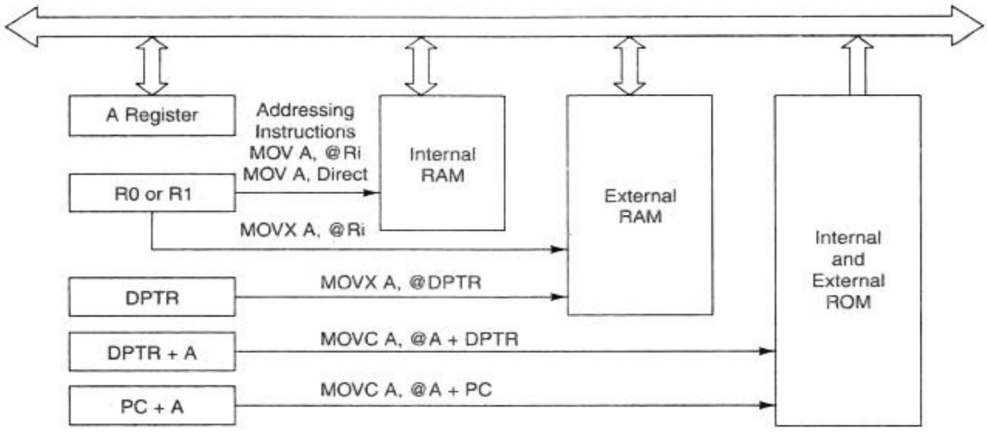


FIG. Addressing Using MOV, MOVX and MOVC

i. Exchange instructions

The content of source ie., register, direct memory or indirect memory will be exchanged with the contents of destination ie., accumulator.

i. XCH A,R3

ii. XCH A,@R1

iii. XCH A,54h

j. Exchange digit. Exchange the lower order nibble of Accumulator (A0-A3) with lower order nibble of the internal RAM location which is indirectly addressed by the register.

i. XCHD A,@R1

ii. XCHD A,@R0

Arithmetic instructions.

The 8051 can perform addition, subtraction. Multiplication and division operations on 8 bit numbers.

Addition

In this group, we have instructions to

i. Add the contents of A with immediate data with or without carry.

i. ADD A, #45H

ii. ADDC A, #OB4H

ii. Add the contents of A with register Rn with or without carry.

i. ADD A, R5

ii. ADDC A, R2

iii. Add the contents of A with contents of memory with or without carry using direct and indirect addressing

i. ADD A, 51H

ii. ADDC A, 75H

iii. ADD A, @R1

iv. ADDC A, @R0

CY AC and OV flags will be affected by this operation.

Subtraction

In this group, we have instructions to

i. Subtract the contents of A with immediate data with or without carry.

i. SUBB A, #45H

ii. SUBB A, #OB4H

ii. Subtract the contents of A with register Rn with or without carry.

i. SUBB A, R5

ii. SUBB A, R2

iii. Subtract the contents of A with contents of memory with or without carry using direct and indirect addressing

i. SUBB A, 51H

ii. SUBB A, 75H

iii. SUBB A, @R1

iv. SUBB A, @R0

CY AC and OV flags will be affected by this operation.

Multiplication

MUL AB. This instruction multiplies two 8 bit unsigned numbers which are stored in A and B register. After multiplication the lower byte of the result will be stored in accumulator and higher byte of result will be stored in B register.

```
Eg. MOV A,#45H ;        [A]=45H
MOV B,#0F5H ;           [B]=F5H
MUL AB ;                [A] x [B] = 45 x F5 = 4209  ;[A]=09H, [B]=42H
```

DIV AB. This instruction divides the 8 bit unsigned number which is stored in A by the 8 bit unsigned number which is stored in B register. After division the result will be stored in accumulator and remainder will be stored in B register.

```
Eg. MOV A,#45H ;        [A]=0E8H
MOV B,#0F5H ;           [B]=1BH
DIV AB ;                [A] / [B] = E8 /1B = 08 H with remainder 10H; [A] = 08H, [B]=10H
```

Increment: *increments the operand by one.*

INC A INC Rn
INC DIRECT
INC @Ri
INC DPTR

INC increments the value of source by 1. If the initial value of register is FFh, incrementing the value will cause it to reset to 0. The Carry Flag is not set when the value "rolls over" from 255 to 0.
In the case of "INC DPTR", the value two-byte unsigned integer value of DPTR is incremented. If the initial value of DPTR is FFFFh, incrementing the value will cause it to reset to 0.

Decrement: *decrements the operand by one.*

DEC A
DEC Rn
DEC DIRECT
DEC @Ri

DEC decrements the value of *source* by 1. If the initial value of is 0, decrementing the value will cause it to reset to FFh. The Carry Flag is not set when the value "rolls over" from 0 to FFh.

Logical Instructions

Logical AND

ANL destination, source: ANL does a bitwise "AND" operation between *source* and *destination*, leaving the resulting value in *destination*. The value in source is not affected. "AND" instruction logically AND the bits of source and destination.

ANL A,#DATA
ANL A, Rn
ANL A,DIRECT
ANL A,@Ri
ANL DIRECT,A
ANL DIRECT, #DATA

Logical OR

ORL destination, source: ORL does a bitwise "OR" operation between *source* and *destination*, leaving the resulting value in *destination*. The value in source is not affected. " OR " instruction logically OR the bits of source and destination.

ORL A,#DATA
ORL A, Rn
ORL A,DIRECT
ORL A,@Ri
ORL DIRECT,A
ORL DIRECT, #DATA

Logical Ex-OR

XRL destination, source: XRL does a bitwise "EX-OR" operation between *source* and *destination*, leaving the resulting value in *destination*. The value in source is not affected. " XRL " instruction logically EX-OR the bits of source and destination.

XRL A,#DATA
XRL A,Rn

XRL A,DIRECT
XRL A,@Ri
XRL DIRECT,A
XRL DIRECT, #DATA

Logical NOT

CPL complements *operand*, leaving the result in *operand*. If *operand* is a single bit then the state of the bit will be reversed. If *operand* is the Accumulator then all the bits in the Accumulator will be reversed.

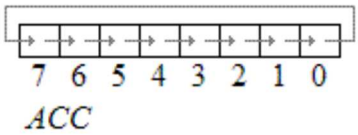
CPL A,
CPL C,
CPL bit address

SWAP A – Swap the upper nibble and lower nibble of A.

Rotate Instructions

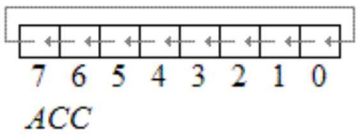
RR A

This instruction is rotate right the accumulator. Its operation is illustrated below. Each bit is shifted one location to the right, with bit 0 going to bit 7.



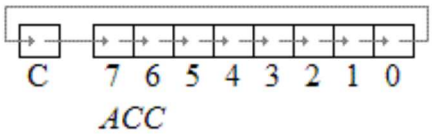
RL A

Rotate left the accumulator. Each bit is shifted one location to the left, with bit 7 going to bit 0



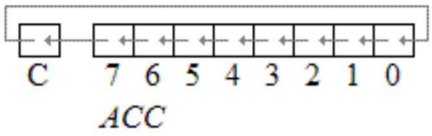
RRC A

Rotate right through the carry. Each bit is shifted one location to the right, with bit 0 going into the carry bit in the PSW, while the carry was at goes into bit 7



RLC A

Rotate left through the carry. Each bit is shifted one location to the left, with bit 7 going into the carry bit in the PSW, while the carry goes into bit 0.



Branch (JUMP) Instructions

Jump and Call Program Range

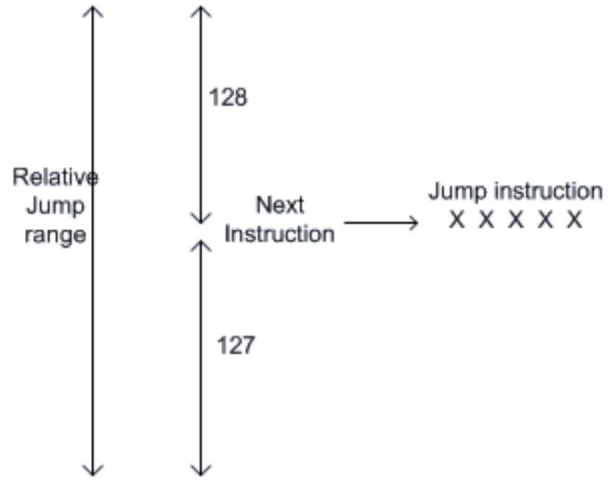
There are 3 types of jump instructions. They are:-

1. Relative Jump
2. Short Absolute Jump
3. Long Absolute Jump

Relative Jump

Jump that replaces the PC (program counter) content with a new address that is greater than (the address following the jump instruction by 127 or less) or less than (the address following the jump by

128 or less) is called a relative jump. Schematically, the relative jump can be shown as follows: -



The advantages of the relative jump are as follows:-
1. Only 1 byte of jump address needs to be specified in the 2's complement form, ie. For jumping ahead, the range is 0 to 127 and for jumping back, the range is -1 to -128.
2. Specifying only one byte reduces the size of the instruction and speeds up program execution.
3. The program with relative jumps can be relocated without reassembling to generate absolute jump addresses.
Disadvantages of the absolute jump: -
1. Short jump range (-128 to 127 from the instruction following the jump instruction)

Instructions that use Relative Jump
SJMP <relative address>; *this is unconditional jump*
The remaining relative jumps are conditional jumps

JC <relative address>
JNC <relative address>
JB bit, <relative address>
JNB bit, <relative address>
JBC bit, <relative address>
CJNE <destination byte>, <source byte>, <relative address>
DJNZ <byte>, <relative address>
JZ <relative address>
JNZ <relative address>

Another classification of jump instructions is
1. Unconditional Jump
2. Conditional Jump

1. **The unconditional jump** is a jump in which control is transferred unconditionally to the target location.
a. **LJMP** (long jump). This is a 3-byte instruction. First byte is the op-code and second and third bytes represent the 16-bit target address which is any memory location from 0000 to FFFFH

eg: LJMP 3000H
b. **AJMP:** this causes unconditional branch to the indicated address, by loading the 11 bit address to 0 -10 bits of the program counter. The destination must be therefore within the same 2K blocks.

c. **SJMP** (short jump). This is a 2-byte instruction. First byte is the op-code and second byte is the relative target address, 00 to FFH (forward +127 and backward -128 bytes from the current PC value). To calculate the target address of a short jump, the second byte is added to the PC value which is address of the instruction immediately below the jump.

2. **Conditional Jump instructions.**

JBC Jump if bit = 1 and clear bit

JNB Jump if bit = 0

JB Jump if bit = 1

JNC Jump if CY = 0

JC Jump if CY = 1

CJNE reg,#data Jump if byte ≠ #data
CJNE A,byte Jump if A ≠ byte
DJNZ Decrement and Jump if A ≠ 0
JNZ Jump if A ≠ 0

JZ Jump if A = 0

All conditional jumps are short jumps.

Bit level jump instructions:

Bit level JUMP instructions will check the conditions of the bit and if condition is true, it jumps to the address specified in the instruction. All the bit jumps are relative jumps.
JB bit, rel ; jump if the direct bit is set to the relative address specified.
JNB bit, rel ; jump if the direct bit is clear to the relative address specified.
JBC bit, rel ; jump if the direct bit is set to the relative address specified and then clear the bit.

The following are the widely used 8051 assembler directives.

ORG (origin)

The ORG directive is used to indicate the starting address. It can be used only when the program counter needs to be changed. The number that comes after ORG can be either in hex or in decimal.
Eg: ORG 0000H ; Set PC to 0000.

EQU and SET

EQU and SET directives assign numerical value or register name to the specified symbol name.
EQU is used to define a constant without storing information in the memory. The symbol defined with EQU should not be redefined.
SET directive allows redefinition of symbols at a later stage.

DB (DEFINE BYTE)

The DB directive is used to define an 8 bit data. DB directive initializes memory with 8 bit values. The numbers can be in decimal, binary, hex or in ASCII formats. For decimal, the 'D' after the decimal number is optional, but for binary and hexadecimal, 'B' and 'H' are required. For ASCII, the number is written in quotation marks ('LIKE This).

END

The END directive signals the end of the assembly module. It indicates the end of the program to the assembler. Any text in the assembly file that appears after the END directive is ignored. If the END statement is missing, the assembler will generate an error message

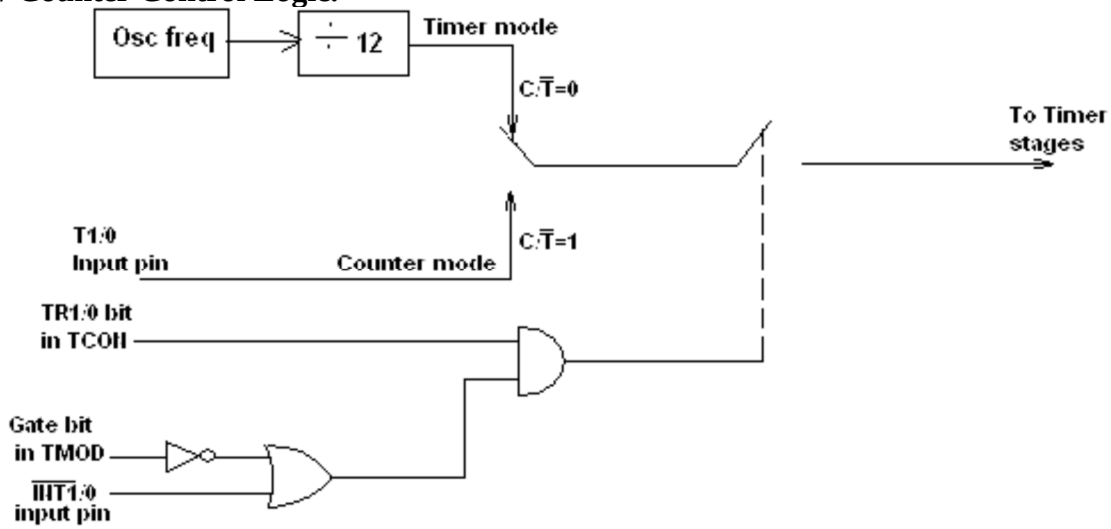
TIMERS AND COUNTERS

Timers/Counters are used generally for

- Time reference
- Creating delay
- Wave form properties measurement
- Periodic interrupt generation
- Waveform generation 8051 has two timers, Timer 0 and Timer 1. Timer in 8051 is used as timer, counter and baud rate generator. Timer always counts up irrespective of whether it is used as timer, counter, or baud rate generator: Timer is always incremented by the microcontroller. The time taken to count one digit up is based on master clock frequency.

If Master CLK=12 MHz, Timer Clock frequency = Master CLK/12 = 1 MHz Timer Clock Period = 1micro second This indicates that one increment in count will take 1 micro second. The two timers in 8051 share two SFRs (TMOD and TCON) which control the timers, and each timer also has two SFRs dedicated solely to itself (TH0/TL0 and TH1/TL1).

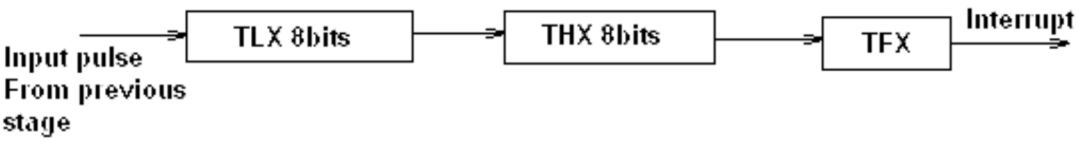
Timer/ Counter Control Logic.



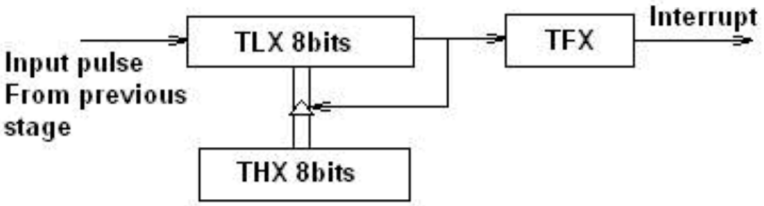
TIMER MODES Timers can operate in four different modes. They are as follows Timer Mode-0: In this mode, the timer is used as a 13-bit UP counter as follows.

Fig. Operation of Timer on Mode-0 The lower 5 bits of TLX and 8 bits of THX are used for the 13 bit count.Upper 3 bits of TLX are ignored. When the counter rolls over from all 0's to all 1's, TFX flag is set and an interrupt is generated. The input pulse is obtained from the previous stage. If TR1/0 bit is 1 and Gate bit is 0, the counter continues counting up. If TR1/0 bit is 1 and Gate bit is 1, then the operation of the counter is controlled by input. This mode is useful to measure the width of a given pulse fed to input.

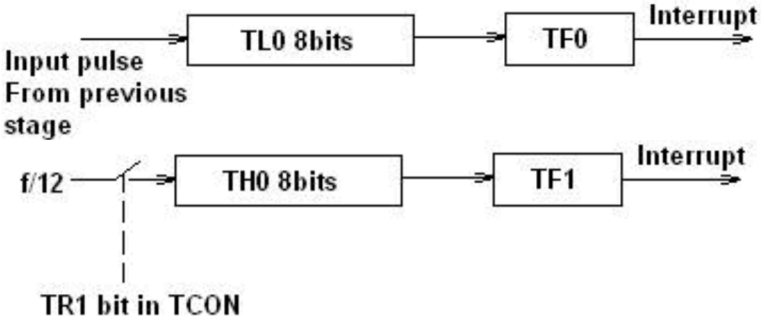
Timer Mode-1: This mode is similar to mode-0 except for the fact that the Timer operates in 16-bit mode.



Timer Mode-2: (Auto-Reload Mode): This is a 8 bit counter/timer operation. Counting is performed in TLX while THX stores a constant value. In this mode when the timer overflows i.e. TLX becomes FFH, it is fed with the value stored in THX. For example if we load THX with 50H then the timer in mode 2 will count from 50H to FFH. After that 50H is again reloaded. This mode is useful in applications like fixed time sampling.



Timer Mode-3: Timer 1 in mode-3 simply holds its count. The effect is same as setting TR1=0. Timer0 in mode-3 establishes TL0 and TH0 as two separate counters.



Control bits TR1 and TF1 are used by Timer-0 (higher 8 bits) (TH0) in Mode-3 while TR0 and TF0 are available to Timer-0 lower 8 bits(TL0).