Techno India NJR Institute of Technology

Department of Electronics & Communication Engineering

B.Tech. IV Semester

Lab: Microcontrollers Lab (4EC4-23)

4EC4-23: Microcontrollers Lab

Credit: 1.5 **Max. Marks: 75(IA:45, ETE:30)**

0L+0T+3P

List of Experiments	
Sr. No.	**Name of Experiment**
Following exercises has to be Performed on 8085	
1.	Write a program for 1.1 Multiplication of two 8 bit numbers 1.2 Division of two 8 bit numbers
2.	Write a program to arrange a set of data in Ascending and Descending order.
3.	Write a program to find Factorial of a given number.
4.	Write a program to generate a Software Delay. 4.1 Using a Register 4.2 Using a Register Pair
8085 Interfacing Programs	
5.	5.1 Write a program to Interface ADC with 8085.
	5.2 Write a program to interface Temperature measurement module with 8085.
6.	Write a program to interface Keyboard with 8085.
7.	Write a program to interface DC Motor and stepper motor with 8085.
Following exercises has to be Performed on 8051	
8.	Write a program to convert a given Hex number to Decimal.
9.	Write a program to find numbers of even numbers and odd numbersamong 10 Numbers.
10.	Write a program to find Largest and Smallest Numbers among 10 Numbers.
11.	11.1 To study how to generate delay with timer and loop.
	11.2 Write a program to generate a signal on output pin using timer.
8051 Interfacing Programs	
12	12.1 Write a program to interface Seven Segment Display with 8051.
	12.2 Write a program to interface LCD with 8051.
13	Write a program for Traffic light Control using 8051.
14	Write a program for Elevator Control using 8051.

Course Outcome:

Course Code	Course Name	Course Outcome	Details
4EC4-23	Microcontrollers Lab	CO 1	Develop skills related to assembly level programming of microprocessors and microcontroller.
		CO 2	Interpret the basic knowledge of microprocessor and microcontroller interfacing, delay generation, waveform generation and Interrupts.
		CO 3	Interfacing the external devices to the microcontroller and microprocessor to solve real time problems.
		CO 4	Illustrate functions of various general purpose interfacing devices.
		CO 5	Develop a simple microcontroller and microprocessor based systems

CO-PO Mapping:

Subject	Course Outcomes	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12
4EC4-23 Microcontrollers Lab	CO 1	2	1	2	1	3							
	CO 2	3	2	1	2	1							
	CO 3	1	1	3	1	3							
	CO 4	2	2	1									
	CO 5	1	1	3	2	2		2					

3: Strongly **2: Moderate** **1: Weak**

Office of Dean Academic Affairs
Rajasthan Technical University, Kota

Course Outcomes:

CO1:	**Comprehension**	Develop skills related to assembly level programming of microprocessors and microcontroller.
CO2:	**Application**	Demonstrate the working of microprocessor and microcontroller by interfacing of pheripheral devices, delay generation, waveform generation and Interrupts calling.
CO3:	**Analysis**	Discriminate microcontroller and microprocessor on the basis of real time problems.
CO4:	**Synthesis**	Develop a simple microcontroller and microprocessor based systems.
CO5:	**Evaluate**	Student will able to Compare the perfomance of diffrent microcontroller familes on the basis response time.

Course Outcome Mapping with Program Outcome:

	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CO1	1	0	2	1	2	2	0	0	0	0	2	2
CO2	1	0	2	0	2	1	0	0	0	0	2	2
CO3	1	2	2	0	1	2	0	0	1	0	2	2
CO4	2	2	1	0	1	2	1	0	1	0	2	2
CO5	0	0	2	2	2	2	1	0	0	0	2	2

INSTRUCTIONS FOR THE LAB

DO'S

1. Student should get the record of previous experiment checked before starting the new experiment.
2. Read the manual carefully before starting the experiment.
3. Before starting the experiment, get circuit diagram checked by the teacher.
4. Before switching on the power supply, get the circuit connections checked.
5. Get your readings checked by the teacher.
6. Apparatus must be handled carefully.
7. Maintain strict discipline.
8. Keep your mobile phone switched off or in vibration mode.
9. Students should get the experiment allotted for next turn, before leaving the lab.

DON'TS

1. Do not touch or attempt to touch the mains power supply wire with bare hands.
2. Do not overcrowd the tables.
3. Do not tamper with equipment's.
4. Do not leave the lab without permission from the teacher.

8085 INTRODUCTION

Aim: Study the hardware, functions, memory structure and operation of 8085-Microprocessor kit.

Apparatus required: VINYTICS VMC-8501 kit

Theory:

SYSTEM INTRODUCTION

VINYTICS VMC-8501 is a single board **MICROPROCESOR WORKSTATION** configured around the most widely used Microprocessor of today's world. Based on 8085 Microprocessor, it can be used to train engineers to control any industrial process and to develop software for 8080 A and 8085 based systems.

GENERAL DESCRIPTION

VMC-8501 kit communicates with the outside world through a keyboard having 64 ASCII keys and Liquid Crystal Display (LCD) of 20*2. The kit also has the capability of interacting with CRT Terminal and Printer through the interfaces provided on the Board. Other devices like floppy drives, etc. can also be connected to the kit through the USART chip 8521.

The VMC-8501 kits have been provided with on board A/D Chips, Relay contacts and Opt Isolators to enable the user to use it for some practical applications.

VMC-8501 provides 8 Kbytes of RAM and 16 Kbytes of EPROM. The total on the board memory can be very easily expanded to 64 Kbytes in an appropriate combination of RAM and ROM.

The Input/output structure of VMC-8501 provides 24 programmable I/O lines expandable to 48 I/O lines. It has got 16 bit programmable Timer/Counter for generating any type of counting etc. The onboard 8529 provides 8 levels of interrupts. The onboard battery backup for RAM retails the memory contents in case of power failure.

The onboard resident system monitor software is very powerful and provides various software utilities. The kit provides various powerful software commands like INSERT, DELETE, BLOCK MOVE, RELOCATE, STRING, FILL, PRINT, PROGRAMMING, VERIFY, LIST & MEMORY COMPARE etc. which are very helpful in debugging/developing the software.

VMC-8501 is configured on the internationally adopted STD Bus, which is the most popular bus for process control nod real time applications.

SYSTEM SPECIFICATION

CPU	8-bit Microprocessor, 8085-A
MEMORY	Total onboard capacity of 64 Kbytes
RAM	16 Kbytes with battery backup, space for further expansion
ROM	32 Kbytes of EPROM loaded with powerful monitor program, space for further expansion using 2732/2764/27128/27256/27512
TIMER	3 Nos. of 16 bit programmable timer/counter using 8253
I/O	48 I/O lines using two 8255
INTERRUPT	Interrupt controller using 8259
KEYBOARD	IBM PC compatible keyboard
LCD DISPLAY	20*2 Alphanumeric Liquid crystal display
BUS	All data ,address and control signals (TTL compatible) available at edge connector
INTERFACE	1) RS-232C through SID/SOD lines with auto baud rate 2) One RS-232 port through 8251 with programmable baud rate 3) EPROM programmer (Optional) 4) Centronics Printer Interface(Optional)
INTERFACE FOR PROCESS CONTROL	1) 8 Analog Input using ADC 0809. 2) One Analog Output using DAC 0800. 3) One relay contact with LED indicator. 4) One opto isolated input with LED indicator. 5) Onboard real time clock (optional). 6) Logic probe indication.

COMMUNCATION	Can be used with window 95/98/Win XP with hyper terminal Application.
POWER SUPPLY	+5V, 1.5A for the kit,
REQUIREMENT	±12V for serial communication. +12V/+24V/+21V, ±5% 100mA for EPROM programmer respectively.
OPERATING TEMP	0 to 50C.

HARDWARE DESCRIPTION

General

The system has got 8085-A as the central processing unit. The clock frequency for the system is 3.07 MHz and is generated from the crystal of 6.14 MHz.

8085 has got 8 data lines and address lines. The lower 8 address lines 8 bit data lines are multiplexed. Since the lower 8 address bits appear on the bus during the first clock cycle of a machine cycle and the 8 bit data appears on the bus during the 2^{nd} & 3^{rd} clock cycles, it becomes necessary to latch the lower 8 address bits during the first clock cycle so that the 16 bits cycle remains available in subsequent cycles . This achieved using a latch 74-LS-373.

Memory

VMC-8501 provides 8k bytes of CMOS RAM using 6264 chip & bytes of EPROM using 27128. The total on board memory can be expanded MP to 64K bytes. The various chips can be used are 2716, 2732, 2764, 27128, 27256, 6116 & 6264 .there are four memory spaces provided on VMC-8501 Each memory space can be defined any address slot from 0000-FFFF depending on the size of memory chip.

Same way memory chip space can be defined to have any of the chips 2716/ 2732/ 2764/ 27128/ 27256/ 6116/ 6264.

I/O Devices

The various I/P chips used in VMC-8501 are 8279, 8255, 8253, 8251 & 8259.The function role of all these are given below:

8279 (PKDI)

8279 is general purpose programmable keyboard and display I/O interface device designed for the use with the 8085 microprocessor. It provides a scanned interface to 64 ASCII key matrixes provided in VMC-8501 and scanned interface for the 20x2 liquid crystal display .8279 has got 16x8 displays **RAM** which can be loaded or interrogated by the CPU. When a key is pressed, its corresponding code is entered in the **FIFO** queue of

8279 & can now be read by the microprocessor. 8279 also refreshes the display RAM automatically.

8255 (PPI)

8255 is a programmable peripheral interface **(PPI)** designed to use with 8085 microprocessor .This basically act as a general purpose I/O device to interface peripheral equipments to the system bus .

It is not necessary to have an external logic to interface with peripheral device since the functional configuration of 8255 is programmed by the system software .it has 3 I/O ports of 8 lines each (PORT-A, PORT-B & PORT-C) port-C cam be divided into 2 ports of 4 lines each named as Port-C upper & Port-C lower . Any I/O combination of port-A, Port-B, Port-C upper & lower can be defined using the appropriate software commands. The port address for these ports are given in chapter-6 .VMC-8501 provides 6 I/O ports of 8 lines each using two 8255 chips.

8253 (PIT)

This chip is a programmable interval Timer/Counter and can be used for the generation of accurate time delays under software control. Various other functions that can be implemented with this chip are Programmable rate generator, even counter, Binary rate multiplier, Real time clock etc. This chip has got three independent 16 bit counters each having a count rate MP to 2 KHz. The first Timer/Counter (i.e. counter 0) is being used for Single Step operation. The second Timer/Counter (i.e. counter 1) is free. The third Timer/Counter (i.e. counter 2) is being used in the single cycle operation. However, the **CLK, GATE & OUT** signals of all the three Timer/Counters are brought out at connector C5.

8251(USART)

This chip is a programmable communication interface and is used as a peripheral device. This device accepts data characters from the CPU in parallel format and then converts them into serial data characters for the CPU this chip will signal the CPU whenever it can accept a new character for transmission or whenever it has received a character from the CPU. The CPU can read the complete status of it at any time. One such chip is used in **VMC 8501** Kit and these can be used for interfacing any serial device. The connection of 8251 is brought at connector C8.

8259 (INTERRUPT CONTROLLER)

The 8259 is a device specifically designed for use in a real time, interrupt driven micro computer system. It manages eight level of request and has built in features for expandability to other 8259's. It is program by system software as an I/O peripheral. A selection of priority modes is available to the programmer so that the manner in which the requests are processed by 8259 can be configured to match his system requirements. The

priority modes can be changed or reconfigured dynamically at any time during the main program. The various signals of this chip are brought out at connector C5.

Liquid Crystal Display

VMC-8501 PROVIDES 20 x 2 Liquid Crystal Displays. LCD stands for **LIQUID CRYSTAL DISPLAY** and is a commonly used O/P device for the micro processor based systems. As compared to the other commonly used O/P device i.e. light emitting Diode (LED) displays, they consume very less power and they don't emit their own light but use ambient light.

The LCD display used hers is CTS-2021 (20 characters into two lines) 1/16 duty, 1/5 bias. It is A 20 character X2 line display, without back light. A character displayed using 5X7 dots format. It is compatible with both for and eight bit microprocessor.

The LCD display has 14 pins. The various signals on them can be classified into three groups, namely power signals, control signals, data signals.

KEYBOARD DESCRIPTION

List of ASCII Keyboard Commands

KEY	DESCRIPTION
L	List a memory block
M	Examine/Modify Memory
E	Enter a memory block
R	Examine/Modify Register
S	Single step
G	Go
B	Block move
I	Insert
D	Delete
N	Insert Data
O	Delete Data
F	Fill
H	Relocate
J	Memory Compare
K	String
P	Print

INTERNAL ARCHITECTURE OF 8085 MICROPROCESSOR

Control Unit

Generates signals within MP to carry out the instruction, which has been decoded. In reality causes certain connections between blocks of the MP to be opened or closed, so that data goes where it is required, and so that ALU operations occur.

Arithmetic Logic Unit

The ALU performs the actual numerical and logic operation such as 'add', 'subtract', 'AND', 'OR', etc. Uses data from memory and from Accumulator to perform arithmetic. Always stores result of operation in Accumulator.

Registers

The 8085/8080A-programming model includes six registers, one accumulator, and one flag register, as shown in Figure. In addition, it has two 16-bit registers: the stack pointer and the program counter. They are described briefly as follows.
The 8085/8080A has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H, and L as shown in the figure. They can be combined as register pairs - BC, DE, and HL - to perform some 16-bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

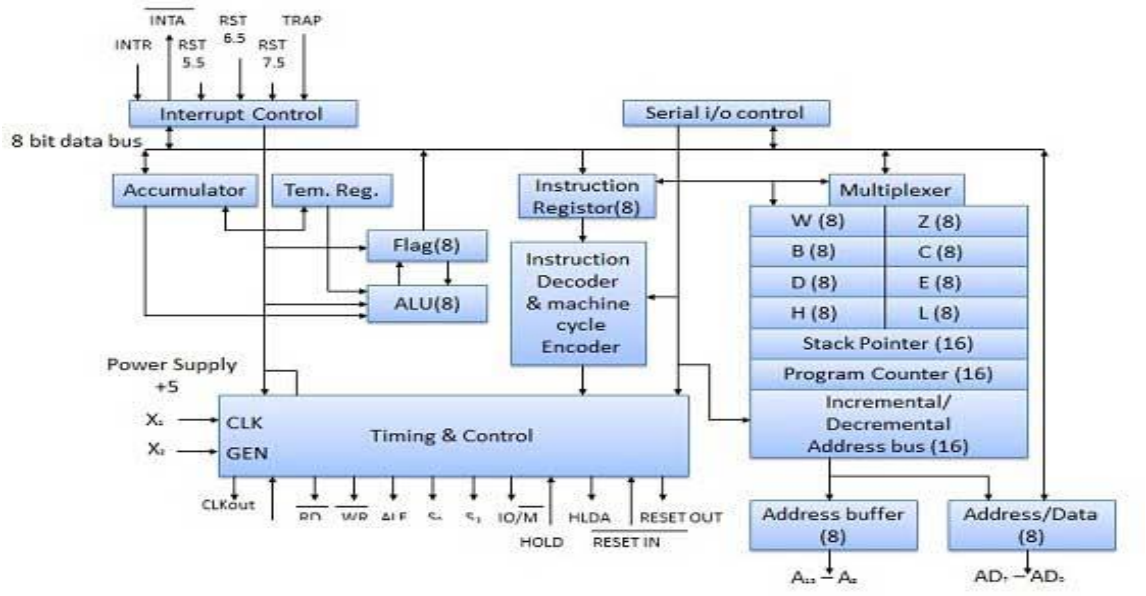

Fig. No. 1:- Internal Architecture of 8085

Accumulator

The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flags

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions.

For example, after an addition of two numbers, if the sum in the accumulator id larger than eight bits, the flip-flop uses to indicate a carry -- called the Carry flag (CY) -- is set to one. When an arithmetic operation results in zero, the flip-flop called the Zero (Z) flag is set to one. The first Figure shows an 8-bit register, called the flag register, adjacent to the accumulator. However, it is not used as a register; five bit positions out of eight are used to store the outputs of the five flip-flops. The flags are stored in the 8-bit register so that the programmer can examine these flags (data conditions) by accessing the register through an instruction.

These flags have critical importance in the decision-making process of the micro-processor. The conditions (set or reset) of the flags are tested through the software instructions. For example, the instruction JC (Jump on Carry) is implemented to change the sequence of a program when CY flag is set. The thorough understanding of flag is essential in writing assembly language programs.

Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register.
The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location

Stack Pointer (SP)

The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer.

Instruction Register/Decoder

Temporary store for the current instruction of a program. Latest instruction sent here from memory prior to execution. Decoder then takes instruction and ‘decodes’ or interprets the instruction. Decoded instruction then passed to next stage.

Memory Address Register
Holds address, received from PC, of next program instruction. Feeds the address bus with addresses of location of the program under execution.

Control Generator
Generates signals within MP to carry out the instruction which has been decoded. In reality causes certain connections between blocks of the MP to be opened or closed, so that data goes where it is required, and so that ALU operations occur.

Register Selector

This block controls the use of the register stack in the example. Just a logic circuit which switches between different registers in the set will receive instructions from Control Unit.

General Purpose Registers
MP requires extra registers for versatility. Can be used to store additional data during a program. More complex processors may have a variety of differently named registers.
Microprogramming

How the MP does knows what an instruction means, especially when it is only a binary number? The micro program in an MP/uC is written by the chip designer and tells the MP/uC the meaning of each instruction MP/uC can then carry out operation.

8085 SYSTEM BUS

Typical system uses a number of buses, collection of wires, which transmit binary numbers, one bit per wire. A typical microprocessor communicates with memory and other devices (input and output) using three buses: Address Bus, Data Bus and Control Bus.

Address Bus

One wire for each bit, therefore 16 bits = 16 wires. Binary number carried alerts memory to 'open' the designated box. Data (binary) can then be put in or taken out. The Address Bus consists of 16 wires, therefore 16 bits. Its "width" is 16 bits. A 16 bit binary number allows 2 16 different numbers, or 32000 different numbers, i.e. 0000000000000000 MP to 1111111111111111. Because memory consists of boxes, each with a unique address, the size of the address bus determines the size of memory, which can be used. To communicate with memory the microprocessor sends an address on the address bus, e.g. 0000000000000011 (3 in decimal), to the memory. The memory the selects box number 3 for reading or writing data. Address bus is unidirectional, i.e. numbers only sent from microprocessor to memory, not other way.

Data Bus

Data Bus: carries 'data', in binary form, between μP and other external units, such as memory. Typical size is 8 or 16 bits. Size determined by size of boxes in memory and μP size helps determine performance of μP. The Data Bus typically consists of 8 wires. Therefore, 2^8 combinations of binary digits. Data bus used to transmit "data", i.e. information, results of arithmetic, etc, between memory and the microprocessor. Bus is bi-directional. Size of the data bus determines what arithmetic can be done. If only 8 bits wide then largest number is 11111111 (255 in decimal). Therefore, larger number has to be broken down into chunks of 255. This slows microprocessor. Data Bus also carries

instructions from memory to the microprocessor. Size of the bus therefore limits the number of possible instructions to 256, each specified by a separate number.

Control Bus

Control Bus is various lines which have specific functions for coordinating and controlling MP operations. E.g.: Read/ Not Write line, single binary digit. Control whether memory is being 'written to' (data stored in mem) or 'read from' (data taken out of mem) 1 = Read, 0 = Write. May also include clock line(s) for timing/ synchronizing, 'interrupts', 'reset' etc. Typically MP has 10 control lines. Cannot function correctly without these vital control signals.

The Control Bus carries control signals partly unidirectional, partly bi directional. Controls signals are things like "read or write". This tells memory that we are reading from a location, specified on the address bus, or writing to a location specified. Various other signals to control and coordinate the operation of the system. Modern day microprocessors, like 80386, 80486 have much larger busses. Typically 16 or 32 bit busses, which allow larger number of instructions, more memory location, and faster arithmetic. Microcontrollers organized along same lines, except: because microcontrollers have memory etc inside the chip, the buses may all be internal. In the microprocessor the three busses are external to the chip (except for the internal data bus). In case of external buses, the chip connects to the busses via buffers, which are simply an electronic connection between external bus and the internal data bus.

8085 PIN DESCRIPTION

Properties

Single + 5V Supply
4 Vectored Interrupts (One is Non Mask able)
Serial In/Serial out Port
Decimal, Binary, and
Double Precision Arithmetic
Direct Addressing Capability to 64K bytes of memory

The Intel 8085A is a new generation, complete 8 bit parallel central processing unit (CPU). The 8085A uses a multiplexed data bus. The address is split between the 8bit address bus and the 8bit data bus.

Pin Description

The following describes the function of each pin:

A8 - A15 (Output 3 State)

Address Bus; The most significant 8 bits of the memory address or the 8 bits of the I/0 addresses, 3 stated during Hold and Halt modes.

AD0 - AD7 (Input/ Output 3state)

Multiplexed Address/Data Bus; Lower 8 bits of the memory address (or I/0 addresses) appear on the bus during the first clock cycle of a machine state. It then becomes the data bus during the second and third clock cycles. 3 stated during Hold and Halt modes.

ALE (Output)

Address Latch Enable: It occurs during the first clock cycle of a machine state and enables the address to get latched into the on chip latch of peripherals. The falling edge of ALE is set to guarantee setup and hold times for the address information. ALE can also be used to strobe the status information. ALE is never stated.

WR (Output 3state)

WRITE; indicates the data on the Data Bus is to be written into the selected memory or I/O location. Data is set MP at the trailing edge of WR. 3 stated during Hold and Halt modes.

READY (Input)

If Ready is high during a read or writes cycle, it indicates that the memory or peripheral is ready to send or receive data. If Ready is low, the CPU will wait for Ready to go high before completing the read or write cycle.

RD (Output 3state)

READ; indicates the selected memory or I/O device is to be read and that the Data Bus is available for the data transfer.

SO, S1 (Output)

Data Bus Status. Encoded status of the bus cycle:

S1	S0	
0	0	HALT
0	1	WRITE
1	0	READ
1	1	FETCH

S1 can be used as an advanced R/W status.

HOLD (Input)

HOLD; indicates that another Master is requesting the use of the Address and Data Buses. The CPU, MP on receiving the Hold request. Will relinquish the use of buses as soon as the completion of the current machine cycle. Internal processing can continue.

The processor can regain the buses only after the Hold is removed. When the Hold is acknowledged, the Address, Data, RD, WR, and IO/M lines are 3stated.

HLDA (Output)

HOLD ACKNOWLEDGE; indicates that the CPU has received the Hold request and that it will relinquish the buses in the next clock cycle. HLDA goes low after the Hold request is removed. The CPU takes the buses one half clock cycle after HLDA goes low.

INTA (Output)

INTERRUPT ACKNOWLEDGE; is used instead of (and has the same timing as) RD during the Instruction cycle after an INTR is accepted. It can be used to activate the 8259 Interrupt chip or some other interrupt port.

RST 5.5
RST 6.5 - (Inputs)
RST 7.5

INTR (Input)

INTERRUPT REQUEST; is used as a general purpose interrupt. It is sampled only during the next to the last clock cycle of the instruction. If it is active, the Program Counter (PC) will be inhibited from incrementing and an INTA will be issued. During this cycle a RESTART or CALL instruction can be inserted to jump to the interrupt service routine. The INTR is enabled and disabled by software. It is disabled by Reset and immediately after an interrupt is accepted.

TRAP (Input)

Trap interrupt is a nonmaskable restart interrupt. It is recognized at the same time as INTR. It is unaffected by any mask or Interrupt Enable. It has the highest priority of any interrupt.

RESTART INTERRUPTS:

These three inputs have the same timing as INTR except they cause an internal RESTART to be automatically inserted.

RST 7.5 ~~ Highest Priority
RST 6.5
RST 5.5 o Lowest Priority

The priority of these interrupts is ordered as shown above. These interrupts have a higher priority than the INTR.

RESET IN (Input)

Reset sets the Program Counter to zero and resets the Interrupt Enable and HLDA flip-flops. None of the other flags or registers (except the instruction register) are affected The CPU is held in the reset condition as long as Reset is applied.

RESET OUT (Output)

Indicates CPU is being reset. Can be used as a system RESET. The signal is synchronized to the processor clock. X1, X2 (Input) Crystal or R/C network connections to set the internal clock generator X1 can also be an external clock input instead of a crystal. The input frequency is divided by 2 to give the internal operating frequency.

CLK (Output)

Clock Output for use as a system clock when a crystal or R/ C network is used as an input to the CPU. The period of CLK is twice the X1, X2 input period.

IO/M (Output)

IO/M indicates whether the Read/Write is to memory or l/O Tristated during Hold and Halt modes.

SID (Input)

Serial input data line the data on this line is loaded into accumulator bit 7 whenever a RIM instruction is executed.

SOD (output)

Serial output data line. The output SOD is set or reset as specified by the SIM instruction.

Vcc

+5 volt supply

Vss

Ground Reference

8085 FUNCTIONAL DESCRIPTION

The 8085A is a complete 8 bit parallel central processor. It requires a single +5 volt supply. Its basic clock speed is 3 MHz thus improving on the present 8080's performance with higher system speed. Also it is designed to fit into a minimum system of three IC's: The CPU, a RAM/ IO, and a ROM or PROM/IO chip.

The 8085A uses a multiplexed Data Bus. The address is split between the higher 8bit Address Bus and the lower 8bit Address/Data Bus. During the first cycle the address is sent out. The lower 8bits are latched into the peripherals by the Address Latch Enable (ALE). During the rest of the machine cycle the Data Bus is used for memory or l/O data.

The 8085A provides RD, WR, and lO/Memory signals for bus control. An Interrupt Acknowledge signal (INTA) is also provided. Hold, Ready, and all Interrupts are synchronized. The 8085A also provides serial input data (SID) and serial output data (SOD) lines for simple serial interface.

In addition to these features, the 8085A has three maskable, restart interrupts and one non maskable trap interrupt. The 8085A provides RD, WR and IO/M signals for Bus control.

Fig. No. 2:- Pin Diagram of 8085

Status Information

Status information is directly available from the 8085A. ALE serves as a status strobe. The status is partially encoded, and provides the user with advanced timing of the type of bus transfer being done. IO/M cycle status signal is provided directly also. Decoded So, S1 Carries the following status information.

Basic System Timing

The 8085A has a multiplexed Data Bus. ALE is used as a strobe to sample the lower 8bits of address on the Data Bus. Figure 2 shows an instruction fetch, memory read and l/ O write cycle (OUT). Note that during the l/O write and read cycle that the l/O port address is copied on both the upper and lower half of the address. As in the 8080, the READY line is used to extend the read and write pulse lengths so that the 8085A can be used with slow memory. Hold causes the CPU to relinquish the bus when it is through with it by floating the Address and Data Buses.

Halt, Write, Read, Fetch

S1 can be interpreted as R/W in all bus transfers. In the 8085A the 8 LSB of address are multiplexed with the data instead of status. The ALE line is used as a strobe to enter the lower half of the address into the memory or peripheral address latch. This also frees extra pins for expanded interrupt capability.

Interrupt and Serial l/O

The8085A has 5 interrupt inputs: INTR, RST5.5, RST6.5, RST 7.5, and TRAP. INTR is identical in function to the 8080 INT. Each of the three RESTART inputs, 5.5, 6.5, 7.5, has a programmable mask. TRAP is also a RESTART interrupt except it is non-maskable.

The three RESTART interrupts cause the internal execution of RST (saving the program counter in the stack and branching to the RESTART address) if the interrupts are enabled and if the interrupt mask is not set. The non-maskable TRAP causes the internal execution of a RST independent of the state of the interrupt enable or masks.

The interrupts are arranged in a fixed priority that determines which interrupt is to be recognized if more than one is pending as follows: TRAP highest priority, RST 7.5, RST 6.5, RST 5.5, INTR lowest priority This priority scheme does not take into account the priority of a routine that was started by a higher priority interrupt. RST 5.5 can interrupt a RST 7.5 routine if the interrupts were re-enabled before the end of the RST 7.5 routine. The TRAP interrupt is useful for catastrophic errors such as power failure or bus error. The TRAP input is recognized just as any other interrupt but has the highest priority. It is not affected by any flag or mask. The TRAP input is both edge and level sensitive.

System Interface

8085A family includes memory components, which are directly compatible to the 8085A CPU. For example, a system consisting of the three chips, 8085A, 8156, and 8355 will have the following features:

2K Bytes ROM
256 Bytes RAM
1 Timer/Counter
4 8bit l/O Ports
1 6bit l/O Port
4 Interrupt Levels
Serial In/Serial out Ports

In addition to standard l/O, the memory mapped I/O offers an efficient l/O addressing technique. With this technique, an area of memory address space is assigned for l/O address, thereby, using the memory address for I/O manipulation. The 8085A CPU can also interface with the standard memory that does not have the multiplexed address/data bus.

The 8085 PROGRAMMING MODEL

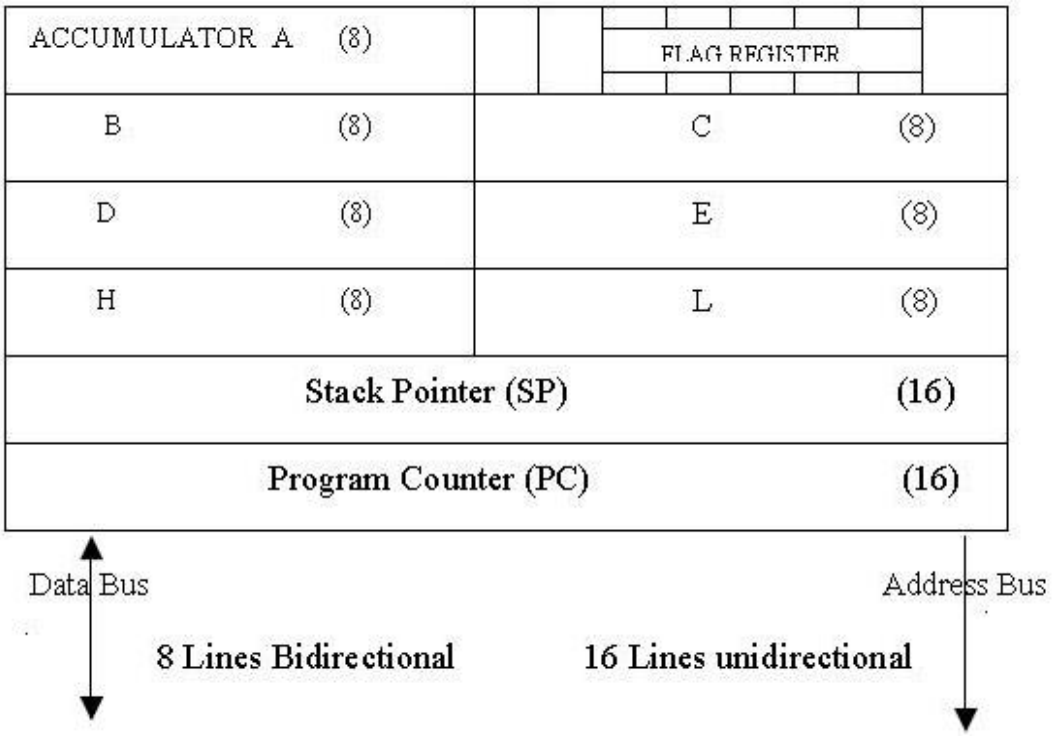

Fig. No.3:- Programming Model of 8085

The 8085 programming model includes six registers, one accumulator, and one flag register, as shown in Figure. In addition, it has two 16-bit registers: the stack pointer and the program counter. They are described briefly as follows.

Registers

The 8085 has six general-purpose registers to store 8-bit data; these are identified as B, C, D, E, H and L as shown in the figure. They can be combined as register pairs - BC, DE, and HL - to perform some 16-bit operations. The programmer can use these registers to store or copy data into the registers by using data copy instructions.

Accumulator

The accumulator is an 8-bit register that is a part of arithmetic/logic unit (ALU). This register is used to store 8-bit data and to perform arithmetic and logical operations. The result of an operation is stored in the accumulator. The accumulator is also identified as register A.

Flags

The ALU includes five flip-flops, which are set or reset after an operation according to data conditions of the result in the accumulator and other registers. They are called Zero (Z), Carry (CY), Sign (S), Parity (P), and Auxiliary Carry (AC) flags; their bit positions in the flag register are shown in the Figure below. The most commonly used flags are Zero, Carry, and Sign. The microprocessor uses these flags to test data conditions.

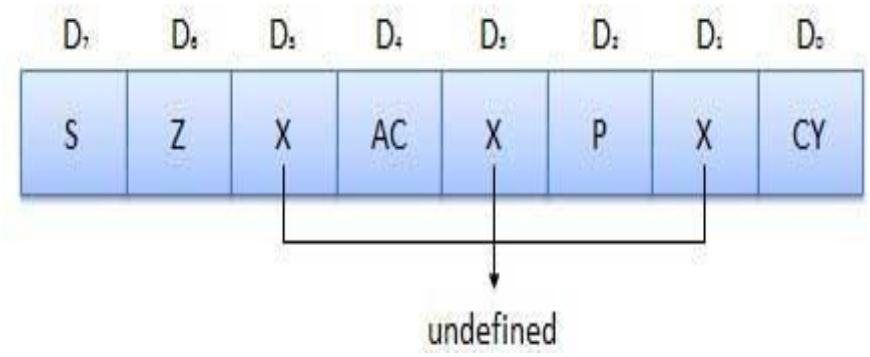

Fig. No. 4:- Flag Register of 8085

For example, after an addition of two numbers, if the sum in the accumulator id larger than eight bits, the flip-flop uses to indicate a carry-- called the Carry flag (CY) -- is set to one. When an arithmetic operation results in zero, the flip-flop called the Zero (Z) flag is set to one. The first Figure shows an 8-bit register, called the flag register, adjacent to the accumulator. However, it is not used as a register; five bit positions out of eight are used to store the outputs of the five flip-flops. The flags are stored in the 8-bit register so that the programmer can examine these flags (data conditions) by accessing the register through an instruction.

These flags have critical importance in the decision-making process of the micro-processor. The conditions (set or reset) of the flags are tested through the software instructions. For example, the instruction JC (Jump on Carry) is implemented to change the sequence of a program when CY flag is set. The thorough understanding of flag is essential in writing assembly language programs.

Program Counter (PC)

This 16-bit register deals with sequencing the execution of instructions. This register is a memory pointer. Memory locations have 16-bit addresses, and that is why this is a 16-bit register. The microprocessor uses this register to sequence the execution of the instructions. The function of the program counter is to point to the memory address from which the next byte is to be fetched. When a byte (machine code) is being fetched, the program counter is incremented by one to point to the next memory location

Stack Pointer (SP)

The stack pointer is also a 16-bit register used as a memory pointer. It points to a memory location in R/W memory, called the stack. The beginning of the stack is defined by loading 16-bit address in the stack pointer. This programming model will be used in subsequent tutorials to examine how these registers are affected after the execution of an instruction.

WORK SPACE

EXP: 1.1

Aim: Write a program for multiplication of two 8 bit numbers (05H, 03H).

Apparatus required: VINYTICS VMC-8501 kit

Procedure:

a) Take two numbers i.e. multiplicand (05H) and multiplier (03H).
b) Take a counter and initialize it as 03.
c) Add 05H with 00H and decrement counter. After counter becomes zero you will get result 0Fh

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	06H	MVI	B,05H	Move data into register.
2001H	05H			
2002H	0EH	MVI	C,03H	Move data into register.
2003H	03H			
2004H	3EH	MVI	A,00H	Move data into accumulator.
2005H	00H			
2006H	80H	ADD	B	Add register B and Accumulator.
2007H	0DH	DCR	C	Decrement in register C.
2008H	C2H	JNZ	2006H	Jump at the location 2006H when C is not equals to zero.
2009H	06H			
200AH	20H			
200BH	32H	STA	2050H	Store the result at the location M2050.
200CH	50H			
200DH	20H			
200EH	76H	HLT		Stop the program.

Output:

BEFORE EXECUTION		AFTER EXECUTION	
MEMORY ADDRESS	CONTENT	MEMORY ADDRESS	CONTENT
M2001H	05H	M2050H	0FH
M2003H	03H		

Result:

We have concluded that how arithmetic operation like multiplication can be implemented using easier addition technique.

Discussion:

1) Why the number of out ports in the peripheral-mapped I/O is restricted to 256 ports?
2) If an input and output port can have the same 8-bit address how does the 8085 differentiate between the ports?
3) Why a latch is used for the output port and a tri-state buffer is used for the input port??
4) Define Memory mapped I/O?
5) Explain RLC, JPO, DI, OUT instructions?

EXP: 1.2

Aim: Write a program for division of two 8 bit numbers (09H, 02H).

Apparatus required: VINYTICS VMC-8501 kit

Procedure:

a) Initialize the Quotient as zero
b) Load the 1st 8 bit data in A
c) Load the 2nd 8 bit data in B
d) Make a counter C = 00.
e) Compare both the values (A,B)
f) Jump if carry to 200FH. (Divisor is greater than dividend) to store result.
g) Otherwise Subtract B from A and increment counter.
h) Jump unconditionally to again compare both the values
i) Jump if carry. (divisor is greater than dividend)
j) Store the results.

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	3EH	MVI	A,09H	Move data 09 into accumulator.
2001H	09H			
2002H	06H	MVI	B,02H	Move data 02 into register B.
2003H	02H			
2004H	0EH	MVI	C,00H	Move data 00 into register C.
2005H	00H			
2006H	B8H	CMP	B	Compare register B with accumulator.
2007H	DAH	JC	200FH	Jump if carry to M 200F
2008H	0FH			
2009H	20H			
200AH	90H	SUB	B	Subtract register B from accumulator
200BH	0CH	INR	C	Increment in register C.
200CH	C3H	JMP	2006H	Jump unconditionally to 2006H
200DH	06H			

200EH	20H			
200FH	32H	STA	2500H	Store the result at the location M2500.
2010H	00H			
2011H	25H			
2012H	79H	MOV	A,C	Move data from register C to A.
2013H	32H	STA	2501H	Store the result at the location M2501.
2014H	01H			
2015H	25H			
2016H	76H	HLT		Stop the program

Output:

BEFORE EXECUTION		AFTER EXECUTION	
MEMORY ADDRESS	CONTENT	MEMORY ADDRESS	CONTENT
M2001H	09H	M2500H	01H (Remainder)
M2003H	02H	M2501H	04H (Quotient)
M2005H	00H		

Result:

We have concluded that how arithmetic operation like division can be implemented using subtraction method.

Discussion:

1) Explain MOV r, M?
2) How many T-states are there in MVI instruction?
3) How many machine cycles are there in XCHG instruction?
4) Explain CNC, CALL, CPE, DAD SP instructions?

EXP: 2.1

Aim: Write a program to arrange a set of data in ascending order (08H, 06H, 01H, 02H, 09H).

Apparatus required: VINYTICS VMC-8501 kit

Procedure:

a) Take five numbers in five different locations.
b) Now take first number in A and second in M, compare & check for carry (if carry than no change otherwise interchange).
c) Here we have to take two counters, one for internal pass and when it is finished outer pass starts.
d) After number of passes (outer counter = 0), we get result at same location in ascending order.

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	16H	MVI	D,04	Move immediate 04H to Resister D
2001H	04			
2002H	21H	LXI	H,2050H	Point location 2050H by resister pair HL
2003H	50H			
2004H	20H			
2005H	0EH	MVI	C,04H	Move immediate 04H to Resister C
2006H	04H			
2007H	7EH	MOV	A,M	Move M to A
2008H	23H	INX	H	Increment reg. pair HL
2009H	BEH	CMP	M	Compare with memory
200AH	DAH	JC	2012	Jump if carry on 2012
200BH	12H			
200CH	20H			
200DH	46H	MOV	B,M	MOV memory to B
200EH	77H	MOV	M,A	MOV memory to A
200FH	2BH	DCX	H	Decrement Reg. Pair HL
2010H	70	MOV	M,B	MOV B to memory
2011H	23H	INX	H	Increment reg. pair HL

2012H	0D	DCR	C	Decrement in C
2013H	C2H	JNZ	2007	Jump if not zero at address 2007H
2014H	07H			
2015H	20H			
2016H	15H	DCR	D	Decrement in D
2017H	C2H	JNZ	2002	Jump if not zero at address 2002H
2018H	02H			
2019H	20H			
201AH	76H	HLT		Stop the program.

Output:

BEFORE EXECUTION		AFTER EXECUTION	
MEMORY ADDRESS	CONTENT	MEMORY ADDRESS	CONTENT
M2050H	08H	M2050H	01H
M2051H	06H	M2051H	02H
M2052H	01H	M2052H	06H
M2053H	02H	M2053H	08H
M2054H	09H	M2054H	09H

Result:

We have concluded that this program gives us practice of using op codes and writing the hex code table with the aim of arranging the data in ascending order.

Discussion:

1) What is microprocessor?
2) Define ROM?
3) Define IC integration stages?
4) What are the four primary operations of a MPU?
5) Explain MOV, MVI, STA, DCR instructions?

EXP: 2.2

Aim: Write a program to arrange a set of data in descending order (08H, 06H, 01H, 02H, 09H).

Apparatus required: VINYTICS VMC-8501 kit

Procedure:

a) Take five numbers in five different locations.
b) Now take first number in A and second in M, compare & check for carry (if carry than interchange otherwise no change).
c) Here we have to take two counters, one for internal pass and when it is finished outer pass starts.
d) After number of passes (outer counter = 0), we get result at same location in descending order.

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	16H	MVI	D,04	Move immediate 04H to Resister D
2001H	04			
2002H	21H	LXI	H,2050H	Point location 2050H by resister pair HL
2003H	50H			
2004H	20H			
2005H	0EH	MVI	C,04H	Move immediate 04H to Resister C
2006H	04H			
2007H	7EH	MOV	A,M	Move M to A
2008H	23H	INX	H	Increment reg. pair HL
2009H	BEH	CMP	M	Compare with memory
200AH	D2H	JNC	2012	Jump if not carry on 2012
200BH	12H			
200CH	20H			
200DH	46H	MOV	B,M	MOV memory to B
200EH	77H	MOV	M,A	MOV memory to A
200FH	2BH	DCX	H	Decrement HL pair
2010H	70	MOV	M,B	MOV B to memory

2011H	23H	INX	H	Increment reg. pair HL
2012H	0D	DCR	C	Decrement in C
2013H	C2H	JNZ	2007	Jump if not zero at address 2007H
2014H	07H			
2015H	20H			
2016H	15H	DCR	D	Decrement in D
2017H	C2H	JNZ	2002	Jump if not zero at address 2002H
2018H	02H			
2019H	20H			
201AH	76H	HLT		Stop the program.

Output:

BEFORE EXECUTION		AFTER EXECUTION	
MEMORY ADDRESS	CONTENT	MEMORY ADDRESS	CONTENT
M2050H	08H	M2050H	09H
M2051H	06H	M2051H	08H
M2052H	01H	M2052H	06H
M2053H	02H	M2053H	02H
M2054H	09H	M2054H	01H

Result:

We have concluded that this program gives us practice of using op codes and writing the hex code table with the aim of arranging the data in descending order.

Discussion:

1) What is an ALU?
2) What do you mean by address bus?
3) How many memory locations can be addressed by a microprocessor with 16 address lines?
4) Why is the data bus bi-directional?
5) Explain INR, DAA, LDA, INX instructions?

EXP: 3

Aim: Write a program to find Factorial of a given number (06H).

Apparatus required: VINYTICS VMC-8501 kit

Procedure:

a) Initialize the stack pointer & Get the number in accumulator
b) Check for if the number is greater than 1. If no store the result otherwise go to next step.
c) Load the counter and initialize result
d) Now factorial program in subroutine is called. In factorial, initialize HL RP with 0.
e) Move the count value to B & Add HL content with Rp
f) Decrement count (for multiplication) & Exchange content of Rp (DE) with HL.
g) Decrement counter (for factorial) till zero flag is set and store the result.

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	3AH	LDA	2550H	Get the number in accumulator from location 2550
2001H	50H			
2002H	25H			
2003H	FEH	CPI	02H	Compare immediate with 02
2004H	02H			
2005H	DAH	JC	201FH	Jump if carry to 201F
2006H	1FH			
2007H	20H			
2008H	16H	MVI	D,00H	Move immediate 00 to reg. D
2009H	00H			
200AH	5FH	MOV	E,A	Move content of A to E
200BH	4FH	MOV	C,A	Move 'A' content to 'C'
200CH	0DH	DCR	C	Decrement register C
200DH	21H	LXI	H,0000H	HL is loaded with data 0000
200EH	00H			
200FH	00H			

2010H	41H	MOV	B,C	Content of 'C' is moved to B
2011H	19H	DAD	D	Content of DE is added with HL
2012H	05H	DCR	B	'B' is decremented
2013H	C2H	JNZ	2011H	Jump if no zero to 2011
2014H	11H			
2015H	20H			
2016H	EBH	XCHG		Exchange [DE] with [HL]
2017H	0DH	DCR	C	Decrement counter value C
2018H	C2H	JNZ	200DH	Jump if no zero to 200D
2019H	0DH			
201AH	20H			
201BH	EB	XCHG		Exchange [DE] with [HL]
201CH	C3	JMP	2022H	Unconditionally Jump to location 2022
201DH	22H			
201EH	20H			
201FH	21H	LXI	H,0001H	HL is loaded with data 01
2020H	00H			
2021H	01H			
2022H	22H	SHLD	2500H	Store content of HL to 2500H
2023H	00H			
2024H	25H			
2025H	76H	HLT		Terminate the program

Output:

BEFORE EXECUTION		AFTER EXECUTION	
MEMORY ADDRESS	CONTENT	MEMORY ADDRESS	CONTENT
2550H	06H	2500H	D0H
		2501H	02H

Result:

We have concluded that how subroutines can be used to find out factorial of a number.

Discussion:

1) Give the three formats of END of Interrupt?
2) What are the signals used by the DMA controller?
3) How long the INTR pulse stays high?
4) List the operating modes of 8255A PPI?
5) Explain RNZ, SBB, SPHL instructions?

EXP: 4.1

Aim: Write a program to generate a software delay of 1 second using a counter made by register.

Apparatus required: VINYTICS VMC-8501 kit, 8255 study card.

Procedure:

a) Initially set value of control word and address of control register, port A. (i.e. 80H, 2BH, 28H).
b) Set value of accumulator first as AAH and 00H. Call delay subroutine for value 0 and value 1.
c) In delay subroutine assign delay by using register method.
d) Connect the 8255 study card with 8085 by using 50 pin for cables.
e) Execute the program and see the result at port A. (LEDs are blinking at particular software delay).

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	3EH	MVI	A,80H	Move immediate 80 to Resister A
2001H	80H			
2002H	D3H	OUT	2BH	Display accumulator on the port address
2003H	2BH			
2004H	3EH	MVI	A,FFH	Move immediate FF to Resister A
2005H	FFH			
2006H	D3H	OUT	28H	Display accumulator on the port address
2007H	28H			
2008H	CDH	CALL	2500H(D)	Call to Delay
2009H	00H			
200AH	25H			
200BH	3EH	MVI	A,00H	Move immediate 00 to Resister A
200CH	00H			
200DH	D3H	OUT	28H	Display accumulator on the port address
200EH	28H			

200FH	CDH	CALL	2500H(D)	Call to Delay
2010H	00H			
2011H	25H			
2012H	C3H	JMP	2004H(S)	Jump unconditionally to 2004H
2013H	04H			
2014H	20H			
2500H	06H	MVI	B,A0H	Move immediate A0 to Resister B
2501H	A0H			
2502H	16H	MVI	D,4AH	Move immediate 4A to Resister D
2503H	4AH			
2504H	00H	NOP		No operation
2505H	00H	NOP		No operation
2506H	00H	NOP		No operation
2507H	00H	NOP		No operation
2508H	15H	DCR	D	Decrement in D
2509H	C2H	JNZ	2504H(N)	Jump if not zero to 2504H
250AH	04H			
250BH	25H			
250CH	05H	DCR	B	Decrement in B
250DH	C2H	JNZ	2502H(M)	Jump if not zero to 2502H
250EH	02H			
250FH	25H			
2510H	C9H	RET		Return to main program

Output: Thus we had used a software way of providing delay to blink LEDs at different intervals.

Result:

We have concluded that how interfacing is implemented practically with the aim of providing software delay 1 sec.

Discussion:

1) List the major components of 8279 keyboard /display interface?
2) What is USART?
3) List the major components of 8251A programmable communication interface?
4) Give the various modes of 8254 timer?
5) Explain PUSH PSW, LXI SP, 16bit, JC, DCX SP instructions?

EXP: 4.2

Aim: Write a program to generate a software delay of 1 second using a counter made by register pair.

Apparatus required: VINYTICS VMC-8501 kit, 8255 study card.

Procedure:

a) Initially set value of control word and address of control register, port A. (i.e. 80H, 2BH, 28H).
b) Set value of accumulator first as AAH and 00H. Call delay subroutine for value 0 and value 1.
c) In delay subroutine assign delay by using register pair method.
d) Connect the 8255 study card with 8085 by using 50 pin for cables.
e) Execute the program and see the result at port A. (LEDs are blinking at particular software delay).

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	3EH	MVI	A,80H	Move immediate 80 to Resister A
2001H	80H			
2002H	D3H	OUT	2BH	Display accumulator on the port address
2003H	2BH			
2004H	3EH	MVI	A,FFH	Move immediate FF to Resister A
2005H	FFH			
2006H	D3H	OUT	28H	Display accumulator on the port address
2007H	28H			
2008H	CDH	CALL	2500H(D)	Call to Delay
2009H	00H			
200AH	25H			
200BH	3EH	MVI	A,00H	Move immediate 00 to Resister A
200CH	00H			
200DH	D3H	OUT	28H	Display accumulator on the port address
200EH	28H			

200FH	CDH	CALL	2500H(D)	Call to Delay
2010H	00H			
2011H	25H			
2012H	C3H	JMP	2004H(S)	Jump unconditionally to 2004H
2013H	04H			
2014H	20H			
2500H	01H	LXI	B,2384H	Load immediate 2384 to Resister Pair B
2501H	84H			
2502H	23H			
2503H	08H	DCX	B	Decrement Register pair
2504H	79H	MOV	A,C	Move content of register C to A
2505H	B0H	ORA	B	Oring of B register content with A
2506H	C2H	JNZ	LOOP	Jump if not zero to 2503H
2507H	03H			
2508H	25H			
2509H	C9H	RET		Return to main program

Output: Thus we had used a software way of providing delay to blink LEDs at different intervals.

Result:

We have concluded that how interfacing is implemented practically with the aim of providing software delay 1 sec.

Discussion:

1) What is the function of program counter?
2) What are different ways of generating delay in 8085?
3) Explain branching instructions used in 8085?
4) Explain Square wave generator mode of 8254?
5) Explain difference between JUMP and CALL instructions?

EXP: 5.1

Aim: Write a program to Interface ADC with 8085.

Apparatus required: VINYTICS VMC-8501 kit, VINYTICS ADC-0809 kit

Procedure:

a) Initially set value of stack pointer and 8255 port.
b) Call delay subroutine between each data set**.**
c) The selected channel at location 2008 input is to be fed at pin Analog input connection and give output as hex value.

00.00 V – 00
01.25 V – 3F
02.50 V – 7F
05.00 V – FF

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	31H	LXI	SP,20FF	Load Stack pointer with 20FF
2001H	FFH			
2002H	20H			
2003H	3EH	MVI	A,98H	INTIALIZE 8255 PORT
2004H	98H			
2005H	D3H	OUT	03	
2006H	03H			
2007H	3EH	MVI	A,00H	Select channel No
2008H	00H			
2009H	D3H	OUT	01	
200AH	01H			
200BH	3EH	MVI	A,00H	
200CH	00H			
200DH	D3H	OUT	02	
200EH	02H			
200FH	3EH	MVI	A,03H	ALE &start of conversion pulse
2010H	03H			
2011H	D3H	OUT	02	
2012H	02H			
2013H	3EH	MVI	A,00H	
2014H	00H			
2015H	D3H	OUT	02	

2016H	02H			
2017H	DBH	IN	02	
2018H	02H			
2019H	E6H	ANI	10	Check EOC
201AH	10H			
201BH	CAH	JZ	LOOP2	
201CH	17H			
201DH	20H			
201EH	3EH	MVI	A,04	Output enable
201FH	04H			
2020H	D3H	OUT	02	
2021H	02H			
2022H	DBH	IN	00	Read ADC data
2023H	00H			
2024H	32H	STA	27F6	
2025H	F6H			
2026H	27H			
2027H	CDH	CALL	CLEAR	Clear display
2028H	47H			
2029H	03H			
202AH	11H	LXI	D,0000	
202BH	00H			
202CH	00H			
202DH	CDH	CALL	DELAY	Call delay
202EH	BCH			
202FH	03H			
2030H	CDH	CALL	MODDOT	Display data
2031H	FAH			
2032H	06H			
2033H	11H	LXI	D,0000	
2034H	00H			
2035H	00H			
2036H	CDH	CALL	DELAY	Call delay
2037H	BCH			
2038H	03H			
2039H	C3H	JMP	LOOP1	Select none sample
203AH	07H			
203BH	20H			

Output: Thus we had used a analog to digital conversion for changing output in digital form and found result as

00.00 V – 00
01.25 V – 3F
02.50 V – 7F

05.00 V – FF

Result: here we had concluded that interfacing of ADC with 8085 make path for analog world to digital world.

Discussion:

1) What are the disadvantages of Analog communication?
2) A 4-bit R/2R digital-to-analog (DAC) converter has a reference of 5 volts. What is the analog output for the input code 0101?

A. 0.3125 V

B. 3.125 V

C. 0.78125 V

D. –3.125 V

3) What are types of ADC's?
4) What are the Advantages of Digital Communication?
5) In a flash analog-to-digital converter, the output of each comparator is connected to an input of a:

A. Decoder

B. priority encoder

C. Multiplexer

D. Demultiplexer

WORK SPACE

EXP: 5.2

Aim: Write a program to interface Temperature measurement module with 8085.

Apparatus required: VINYTICS VMC-8501 kit, VINYTICS VMC-TEMP Temperature Controller interface

Procedure:

d) Connect +5V, +12V, -12V and GND to the module.
e) Connect 26 pin FRC Cable from 8255-I of the kit to the module as per the polarity. Connect thermocouple according to the color code.
f) Short Jumpers accordingly:
   (i) SOC – 2-3 for internal SoC
   (ii) EOC – 1-2 for internal EoC
   (iii) O/P Enable – 1-2 for internal OE
   (iv) Clock – 2-3 for internal clock
   (v) Short X4 – for 250 KHz internal clock
g) Enter the program from 2000 memory address of the kit.
h) Execute the program from 2000 memory address of the kit.
i) Apply the temperature to the thermocouple & see the result on display of kit.
   (The program given below is in the monitor of kit and the location is F800:0600. This program is to be block move at the location 0000:0400)

Program:

M.ADDRESS	HEX. CODE	MNEMONICS	
		OPCODE	OPERAND
2000H	3E	MVI	A,98H
2001H	98		
2002H	D3	OUT	03H
2003H	03		
2004H	3E	MVI	A,0H
2005H	00		
2006H	D3	OUT	01H
2007H	01		
2008H	06	MVI	B,00H
2009H	00		
200AH	0E	MVI	C,00H
200BH	00		
200CH	16	MVI	D,0FFH
200DH	FF		
200EH	3E	MVI	A,00H
200FH	00		
2010H	D3	OUT	02H
2011H	02		
2012H	3E	MVI	A,03H
2013H	03		

2014H	D3	OUT	02H
2015H	02		
2016H	3E	MVI	A,00H
2017H	00		
2018H	D3	OUT	02H
2019H	02		
201AH	DB	IN	02H
201BH	02		
201CH	E6	ANI	10H
201DH	10		
201EH	CA	JZ	LOOP1
201FH	1A		
2020H	20		
2021H	3E	MVI	A,04H
2022H	04		
2023H	D3	OUT	02H
2024H	02		
2025H	DB	IN	00H
2026H	00		
2027H	81	ADD	C
2028H	4F	MOV	C,A
2029H	DA	JC	LOOP2
202AH	51		
202BH	20		
202CH	15	DCR	D
202DH	C2	JNZ	LOOP
202EH	0E		
202FH	20		
2030H	37	STC	
2031H	3F	CMC	
2032H	16	MVI	D,08H
2033H	08		
2034H	78	MOV	A,B
2035H	0F	RRC	
2036H	47	MOV	B,A
2037H	79	MOV	A,C
2038H	1F	RAR	
2039H	4F	MOV	C,A
203AH	15	DCR	D
203BH	C2	JNZ	LOOP3
203CH	34		
203DH	20		
203EH	26	MVI	H,25H
203FH	25		
2040H	69	MOV	L,C
2041H	7E	MOV	A,M
2042H	32	STA	27F6H

2043H	F6		
2044H	27		
2045H	CD	CALL	06FAH
2046H	FA		
2047H	06		
2048H	11	LXI	D,9040H
2049H	40		
204AH	90		
204BH	CD	CALL	03BCH
204CH	BC		
204DH	03		
204EH	C3	JMP	LOOP5
204FH	08		
2050H	20		
2051H	04	INR	B
2052H	C3	JMP	LOOP4
2053H	2C		
2054H	20		
2500H		ORG	2500H
2500H	00 00 01 01 02 02 03 03 04 04 05 05	DFB	00H,00H,01H,01H,02H,02H, 03H,03H,04H,04H,05H,05H
250CH	06 06 07 07 08 08 09 09 10 10 11 11	DFB	06H,06H,07H,07H,08H,08H, 09H,09H,10H,10H,11H,11H
2518H	12 12 13 13 14 14 15 15 16 16 17 17	DFB	12H,12H,13H,13H,14H,14H, 15H,15H,16H,16H,17H,17H
2524H	18 18 19 19 20 20 21 21 22 22 23 23	DFB	18H,18H,19H,19H,20H,20H, 21H,21H,22H,22H,23H,23H
2530H	24 24 25 25 26 26 27 27 28 28 29 29	DFB	24H,24H,25H,25H,26H,26H, 27H,27H,28H,28H,29H,29H
253CH	30 30 31 31 32 32 33 33 34 34 35 35	DFB	30H,30H,31H,31H,32H,32H, 33H,33H,34H,34H,35H,35H
2548H	36 36 37 37 38 38 39 39 39 40 40 40	DFB	36H,36H,37H,37H,38H,38H, 39H,39H,39H,40H,40H,40H

2554H	41 41 41 42 42 42 43 43 43 44 44 44	DFB	41H,41H,41H,42H,42H,42H, 43H,43H,43H,44H,44H,44H
2560H	45 45 45 46 46 46 47 47 47 48 48 48	DFB	45H,45H,45H,46H,46H,46H, 47H,47H,47H,48H,48H,48H
256CH	49 49 49 50 50 50 51 51 51 52 52 52	DFB	49H,49H,49H,50H,50H,50H, 51H,51H,51H,52H,52H,52H
2578H	53 53 53 54 54 54 55 55 55 56 56 56	DFB	53H,53H,53H,54H,54H,54H, 55H,55H,55H,56H,56H,56H
2584H	57 57 57 58 58 58 59 59 59 60 60 60	DFB	57H,57H,57H,58H,58H,58H, 59H,59H,59H,60H,60H,60H
2590H	61 61 61 62 62 62 63 63 64 64 65 65	DFB	61H,61H,61H,62H,62H,62H, 63H,63H,64H,64H,65H,65H
259CH	66 66 67 67 68 68 69 69 69 70 70 71	DFB	66H,66H,67H,67H,68H,68H, 69H,69H,69H,70H,70H,71H
25A8H	71 72 72 73 73 73 74 74 74 75 75 75	DFB	71H,72H,72H,73H,73H,73H, 74H,74H,74H,75H,75H,75H
25B4H	76 76 76 77 77 77 78 78 78 79 79 79	DFB	76H,76H,76H,77H,77H,77H, 78H,78H,78H,79H,79H,79H
25C0H	80 80 80 81 81 81 82 82 82 83 83 83	DFB	80H,80H,80H,81H,81H,81H, 82H,82H,82H,83H,83H,83H
25CCH	84 84 84 85 85 85 86 86 86 87 87 87	DFB	84H,84H,84H,85H,85H,85H, 86H,86H,86H,87H,87H,87H
25D8H	88 88 88 89 89 89 90 90 90 91 91 91	DFB	88H,88H,88H,89H,89H,89H, 90H,90H,90H,91H,91H,91H

25E4H	92 92 92 93 93 93 94 94 94 95 95 95	DFB	92H,92H,92H,93H,93H,93H, 94H,94H,94H,95H,95H,95H
25F0H	96 96 96 97 97 97 97 98 98 98 98 99	DFB	96H,96H,96H,97H,97H,97H, 97H,98H,98H,98H,98H,98H
25FCH	99 99 99 99	DFB	99H,99H,99H,99H

Output: Thus we had used a thermocouple transducer card for measuring temperature. It display the counts after digital conversion.

Result: here we had concluded that interfacing of temperature sensor card with 8085 make life easy as we can see digital outputs for analog temperature values.

Discussion

1) Explain ADC 0808 interface card.
2) What are temperature sensor? Explain different type of temperature sensors with diagrams.
3) Explain magnitude comparator.
4) Explain in details application of automatic temperature control system.

EXP: 6

Aim: Write a program to interface Keyboard with 8085.

Apparatus required: VINYTICS VMC-8501 kit, VINYETICS Keyboard Interface Module

Theory: This program displays the code of the key which is pressed on the keyboard pad. The code is displayed in the data field and remains unchanged till the next key is pressed.

The port of 8255 i.e. P1 is initialized to make port A as input port and port c as output port. The three rows of the key are scanned one by one and process is repeated till the key is pressed till the key is pressed, in the routine code and F code (Final code). The information of code is then displayed and the monitor jumps back again to see if any other key is pressed.

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	3EH	MVI	A,90H	Initialize the port A as an input & port B and C as an output
2001H	90H			
2002H	D3H	OUT	03	
2003H	03H			
2004H	06H	MVI	B,00H	Initialize the final key code in Reg. B
2005H	00H			
2006H	0EH	MVI	C,01H	Put the walking one pattern in reg C with one in LSB position
2007H	01H			
2008H	79H	MOV	A,C	Move the pattern in A
2009H	D3H	OUT	PC	Output of port C
200AH	02H			
200BH	DBH	IN	PA	Input port A
200CH	00H			

200DH	CDH	CALL	CODE	
200EH	38H			
200FH	20H			
2010H	FEH	CPI	08H	Any key closure
2011H	08H			
2012H	FAH	JM	DISP	Yes, Go to display it
2013H	24H			
2014H	20H			
2015H	78H	MOV	A,B	No, move partial result in A
2016H	C6H	ADI	08H	Increment the PC code in partial result
2017H	08H			
2018H	47H	MOV	B,A	
2019H	FEH	CPI	18H	Has PC code become II
201AH	18H			
201BH	F2H	JP	INIT	Yes, Go start scanning from row 0
201CH	04H			
201DH	20H			
201EH	79H	MOV	A,C	
201FH	07H			
2020H	4FH	MOV	C,A	
2021H	C3H	JMP	SCAN	Continue scanning
2022H	08H			
2023H	20H			
2024H	B0H	ORA	B	Or the PA Code with PC Code
2025H	32H	STA	CURDT	Let it be current data
2026H	F6H			
2027H	27H			
2028H	3EH	MVI	A,00	Arg.-No dot
2029H	00H			

202AH	CDH	CALL	UPDDT	Display it in data field
202BH	FAH			
202CH	06H			
202DH	C3H	JMP	INIT	Go to scan the KB again
202EH	04H			
202FH	20H			
2038H	B7H	ORA	A	
2039H	C2H	JNZ	CODE2	
203AH	3FH			
203BH	20H			
203CH	3EH	MVI	A,08	
203DH	08H			
203EH	C9H	RET		
203FH	16H	MVI	D,00	
2040H	00H			
2041H	0FH	RRC		Let LSB in A go to carry
2042H	DAH	JC	CODE10	Go to return if the bit was one increment counter
2043H	49H			
2044H	20H			
2045H	14H	INR	D	Increment counter
2046H	C3H	JMP	CODE5	Check the next bit
2047H	41H			
2048H	20H			
2049H	7AH	MOV	A,D	
204AH	C9H	RET		

Output: Thus we had interfaced keyboard with 8085 and seen that the letter pressed in 4x4 matric keyboard with be displayed at seven segments of 8085.

Result: We have concluded that how interfacing is implemented practically with the aim of providing keyboard interfacing that can be used to give inputs.

Discussion:

1) How a Keyboard matrix is formed in keyboard interface using 8279?
2) What are the tasks involved in keyboard interface?
3) What is scanning in keyboard and what is scan time?
4) What is a programmable peripheral device?

EXP: 7.1

Aim: Write a program to interface DC motor with 8085.

Apparatus required: VINYTICS VMC-8501 kit, VINYTICS DC motor speed controller card

Components required: 12 V AC transformers

Procedure:

a) Connect the transformer secondary to tag no's 6 and 5 of J3 (Vtg. = 12V) for the tapping of the secondary (AC voltage of DC motor)
b) Connect the DC motor to tag's 2 and 3 of J3.
c) Connect the 26 pin FRC connector from the kit to the module and check the polarity of the cable.
d) Connect the power supply to the kit and the module.
e) Enter the program.
f) Observe the motor. Bring the required changes in the program to bring about speed as well as directional changes.
g) If output 'FD' on port C, instead of 'FF', the motor will start rotating in the counter clockwise direction.
h) If 'FF' is output on port A, speed achieved will be maximum and if '00' is output on port A, the motor will stop altogether. However '7F' represents half the speed.

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	3EH	MVI	A,80H	CWR (8255)
2001H	80H			
2002H	D3H	OUT	CWR	
2003H	03H			
2004H	3EH	MVI	A,FFH	Reset all port bits
2005H	00H			
2006H	D3H	OUT	Port A	
2007H	02H			
2008H	3EH	MVI	A,FFH	Set PC2 (PC2=1)
2009H	04H			
200AH	D3H	OUT	Port C	
200BH	02H			
200CH	EFH	RST	5	

Output:

We had successfully controlled the speed of a DC motor in clockwise and anti clock wise direction and observe that motor is rotating in full, zero and half speed.

Result:

We have concluded that we can control speed of a motor and also rotate it clockwise or anti clock wise direction.

Discussion:

1) Which are the important interfacing chips of a microprocessor?
2) What are the sequences of operations that occur when CALL & RET instructions are executed?
3) What is the difference between memory mapped I/O and I/O mapped I/O?
4) Explain transformer with diagram?

EXP: 7.2

Aim: Write a program to interface steeper motor with 8085.

Apparatus required: VINYTICS VMC-8501 kit, Steeper motor controller card

Components required: Steeper Motor

Procedure:

i) Initially set value of control word and address of control register, port A. (i.e. 80H, 2BH, 28H).
j) Set value of accumulator first as F9H, F5H, F6H and FAH for reverse rotation and then observe rotation. To rotate in forward direction reverse the data set.
k) Call delay subroutine between each data set, in delay subroutine assign delay by using nested method**.**
l) Connect the 8255 study card with 8085 by using 50 pin frc cable and steeper motor card by using 24 pin for cables.
m) Connect the steeper motor with card and observe the step movement.
n) For fast movement change the data from 0000H to 0020H.

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	3EH	MVI	A,80H	Move immediate 80 to Resister A
2001H	80H			
2002H	D3H	OUT	03H	Display accumulator on the port address
2003H	03H			
2004H	3EH	MVI	A,FAH	Move immediate FA to Resister A
2005H	FAH			
2006H	D3H	OUT	00H	Display accumulator on the port address
2007H	00H			
2008H	CDH	CALL	2030H(D)	Call to Delay
2009H	30H			
200AH	20H			
200BH	3EH	MVI	A,F6H	Move immediate 00 to Resister A
200CH	F6H			
200DH	D3H	OUT	00H	Display

200EH	00H			accumulator on the port address
200FH	CDH	CALL	2030H(D)	Call to Delay
2010H	30H			
2011H	20H			
2012H	3EH	MVI	A,F5	Move immediate F5 to Resister A
2013H	F5H			
2014H	D3H	OUT	00H	Display accumulator on the port address
2015H	00H			
2016H	CDH	CALL	2030H(D)	Call to Delay
2017H	30H			
2018H	20H			
2019H	3EH	MVI	A,F9H	Move immediate F9 to Resister A
201AH	F9H			
201BH	D3H	OUT	00H	Display accumulator on the port address
201CH	00H			
201DH	CDH	CALL	2030H(D)	Call to Delay
201EH	30H			
201FH	20H			
2020H	C3H	JMP	2004H(S)	Jump unconditionally to 2004H
2021H	04H			
2022H	20H			
2023H	76H	HLT		Stop the program
2030H	11H	LXI	D,0000H	Initialize seconds, minutes
2031H	00H			
2032H	00H			
2033H	CDH	CALL	03BC	Call to Delay
2034H	BCh			
2035H	03H			
2036H	C9H	RET		

Output:

We had successfully controlled the speed of a stepper motor in forward and reverse direction and observe that motor is rotating in steps of 45 degrees.

Result:

We have concluded that we can control speed of a motor and also rotate it forward or reverse direction.

Discussion:

1) List the flags of 8085?
2) Why data bus is Bi-directional?
3) What are programmable peripheral devices?
4) When the 8085 checks for an interrupt?
5) Explain RST 5, RRC, SUB D, SBI instructions?

8051 MICROCONTROLLER INTRODUCTION

Aim: Study the hardware, functions, memory structure and operation of 8051-Microcontroller.

Apparatus required: VINYTICS VMC-8051 kit

Theory:

Earlier to Microcontrollers, Microprocessors were greatly used for each and every purpose. Microprocessors were containing ALU, general purpose register, stack pointer, program counter, clock counter and so many other features which the today's Microcontroller also possesses. But the difference between them exists with respect to the number of instructions, access times, size, reliability, PCB size and so on. Microprocessor contains large instruction set called as CISC processor whereas Microcontroller contains less number of instructions and is called as RISC processor. The access time is less in case of microcontrollers compared to microprocessors and the PCB size reduces in case of microcontrollers.

There are many versions of microcontrollers 8051, 80528751, AT8951 from Atmel Corporation and many more. In this manual we will study about the 8051 architecture, its features, programming and interfacing.

MCS 8051 is an 8-bit single chip microcontroller with many built-in functions and is the core for all MCS-51 devices.

The main features of the 8051 core are:

- Operates with single Power Supply +5V.
- 8-bit CPU optimized for control applications.
- 16-bit program counter (PC) and 16-bit data pointer (DPTR).
- 8-bit program status word (PSW).
- 8-bit stack pointer (SP).
- 4K Bytes of On-Chip Program Memory (Internal ROM or EPROM).
- 128 bytes of On-Chip Data Memory (Internal RAM):
- Four Register Banks, each containing 8 registers (R0 to R7) [Total 32 reg]
- 16-bytes of bit addressable memory.
- 80 bytes of general-purpose data memory (Scratch Pad Area).
- Special Function Registers (SFR) to configure/operate microcontroller.
- 32 bit bi-directional I/O Lines (4 ports P0 to P3).
- Two 16-bit timers/counters (T0 and T1).
- Full duplex UART (Universal Asynchronous Receiver/Transmitter).
- On-Chip oscillator and clock circuitry.

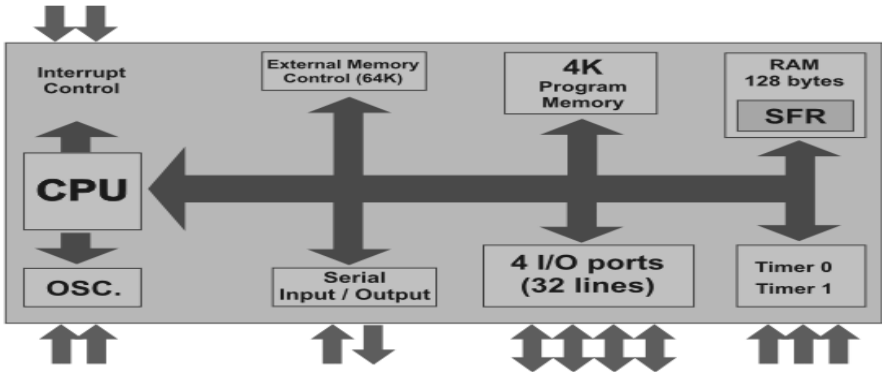

Fig. No. 1: - Architecture of 8051 Microcontroller

Architecture of 8051 microcontroller has following features:-

- 4 Kb of ROM is not much at all.
- 128Kb of RAM (including SFRs) satisfies the user's basic needs.
- 4 ports having in total of 32 input/output lines are in most cases sufficient to make all necessary connections to peripheral environment.

The whole configuration is obviously thought of as to satisfy the needs of most programmers working on development of automation devices. One of its advantages is that nothing is missing and nothing is too much. In other words, it is created exactly in accordance to the average user‘s taste and needs. Other advantages are RAM organization, the operation of Central Processor Unit (CPU) and ports which completely use all recourses and enable further upgrade.

8051 PIN DESCRIPTION

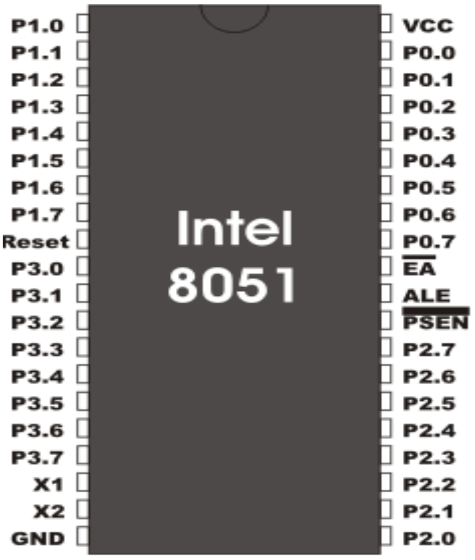

Fig. No. 2: - 8051 Pin Description

- 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 inputs.
- 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: X2, X1 Internal oscillator input and output. A quartz crystal which specifies operating frequency is usually connected to these pins. Instead of it, miniature ceramics resonators can also be used for frequency stability. Later versions of microcontrollers operate at a frequency of 0 Hz up to over 50 Hz.
- 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.

INPUT/OUTPUT PORTS (I/O PORTS)

All 8051 microcontrollers have 4 I/O ports each comprising 8 bits which can be configured as inputs or outputs. Accordingly, in total of 32 input/output pins enabling the microcontroller to be connected to peripheral devices are available for use.
Pin configuration, i.e. whether it is to be configured as an input (1) or an output (0), depends on its logic state. In order to configure a microcontroller pin as an input, it is necessary to apply logic zero (0) to appropriate I/O port bit. In this case, voltage level on appropriate pin will be 0.
Similarly, in order to configure a microcontroller pin as an input, it is necessary to apply a logic one (1) to appropriate port. In this case, voltage level on appropriate pin will be 5V (as is the case with any TTL input).

MEMORY ORGANIZATION

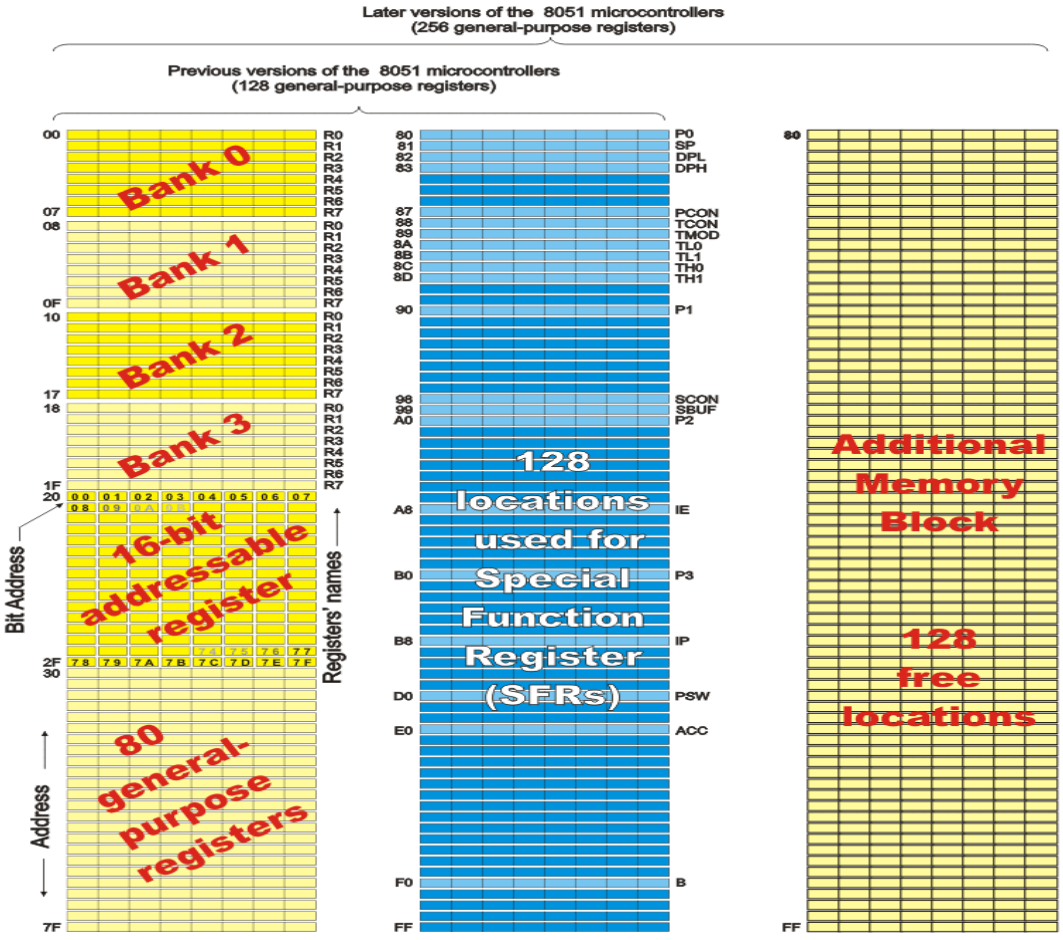

Fig. No. 3: - 8051 Memory Organization

The 8051 has two types of memory and these are 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. Depending on the model in use (we are still talking about the 8051 microcontroller family in general) at most a few Kb of ROM and 128 or 256 bytes of RAM is used.

All 8051 microcontrollers have a 16-bit addressing bus and are capable of addressing 64 kb memory.

SPECIAL FUNCTION REGISTERS (SFRs)

Special Function Registers (SFRs) are a sort of control table used for running and monitoring the operation of the microcontroller. Each of these registers as well as each bit they include, has its name, address in the scope of RAM and precisely defined purpose such as timer control, interrupt control, serial communication control etc. Even though there are 128 memory locations intended to be occupied by them, the basic core, shared by all types of 8051 microcontrollers, has only 21 such registers. Rest of locations is intentionally left unoccupied in order to enable the manufacturers to further develop microcontrollers keeping them compatible with the previous versions. It also enables programs written a long time ago for microcontrollers which are out of production now to be used today.

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8									CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

Bit-addressable Registers

Fig. No. 4: - 8051 Special Function Register

PROGRAM STATUS WORD (PSW) REGISTER

	0	0	0	0	0	0	0	0	Value after Reset
PSW	CY	AC	F0	RS1	RS0	OV		P	Bit name
	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	

Fig. No. 5: - Program Status Word Register

PSW register is one of the most important SFRs. It contains several status bits that reflect the current state of the CPU. Besides, this register contains Carry bit, Auxiliary Carry, two register bank select bits, Overflow flag, parity bit and user-definable status flag.

- P - Parity bit. If a number stored in the accumulator is even then this bit will be automatically set (1), otherwise it will be cleared (0). It is mainly used during data transmit and receive via serial communication.
- Bit 1. This bit is intended to be used in the future versions of microcontrollers.
- OV Overflow occurs when the result of an arithmetical operation is larger than 255 and cannot be stored in one register. Overflow condition causes the OV bit to be set (1). Otherwise, it will be cleared (0).
- RS0, RS1 - Register bank select bits. These two bits are used to select one of four register banks of RAM. By setting and clearing these bits, registers R0-R7 are stored in one of four banks of RAM.

RS1	RS2	Space in RAM
0	0	Bank0 00h-07h
0	1	Bank1 08h-0Fh
1	0	Bank2 10h-17h
1	1	Bank3 18h-1Fh

Fig. No. 6: - Register Bank

- F0 - Flag 0. This is a general-purpose bit available for use.
- AC - Auxiliary Carry Flag is used for BCD operations only.
- CY - Carry Flag is the (ninth) auxiliary bit used for all arithmetical operations and shift instructions.

DATA POINTER REGISTER (DPTR)

DPTR register is not a true one because it doesn't physically exist. It consists of two separate registers: DPH (Data Pointer High) and (Data Pointer Low).

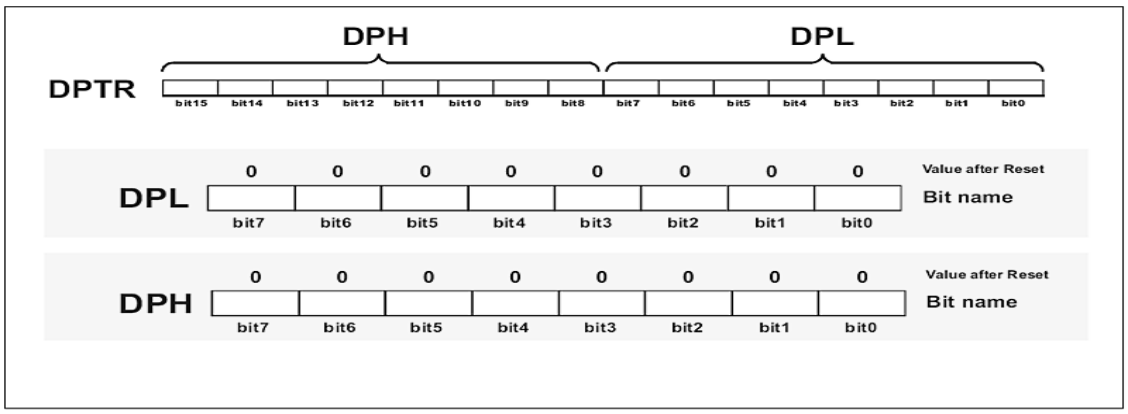

Fig. No. 7: - Data Pointer Register

For this reason, it may be treated as a 16-bit register or as two independent 8-bit registers. Their 16 bits are primarily used for external memory addressing. Besides, the DPTR Register is usually used for storing data and intermediate results.

STACK POINTER (SP) REGISTER

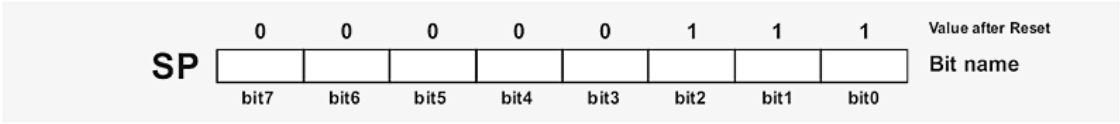

Fig. No. 8: - Stack Pointer Register

A value stored in the Stack Pointer points to the first free stack address and permits stack availability. Stack pushes increment the value in the Stack Pointer by 1. Likewise, stack pops decrement its value by 1. Upon any reset and power-on, the value 7 is stored in the Stack Pointer, which means that the space of RAM reserved for the stack starts at this location. If another value is written to this register, the entire Stack is moved to the new memory location.

P0, P1, P2, P3 - INPUT/OUTPUT REGISTERS

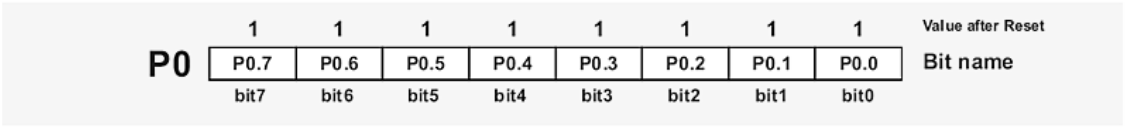

Fig. No. 9: - Input/ Output Register

If neither external memory nor serial communication system are used then 4 ports within total of 32 input/output pins are available for connection to peripheral environment. Each bit within these ports affects the state and performance of appropriate pin of the microcontroller. Thus, bit logic state is reflected on appropriate pin as a voltage (0 or 5 V) and vice versa, voltage on a pin reflects the state of appropriate port bit.
As mentioned, port bit state affects performance of port pins, i.e. whether they will be configured as inputs or outputs. If a bit is cleared (0), the appropriate pin will be configured as an output, while if it is set (1), the appropriate pin will be configured as an input. Upon reset and power-on, all port bits are set (1), which means that all appropriate pins will be configured as inputs.

COUNTERS AND TIMERS

As you already know, the microcontroller oscillator uses quartz crystal for its operation. As the frequency of this oscillator is precisely defined and very stable, pulses it generates are always of the same width, which makes them ideal for time measurement. Such crystals are also used in quartz watches. In order to measure time between two events it is sufficient to count up pulses coming from this oscillator. That is exactly what the timer does. If the timer is properly programmed, the value stored in its register will be incremented (or decremented) with each coming pulse, i.e. once per each machine cycle. A single machine-cycle instruction lasts for 12 quartz oscillator periods, which means that by embedding quartz with oscillator frequency of 12MHz, a number stored in the timer register will be changed million times per second, i.e. each microsecond.

The 8051 microcontroller has 2 timers/counters called T0 and T1. As their names suggest, their main purpose is to measure time and count external events. Besides, they can be used for generating clock pulses to be used in serial communication, so called Baud Rate.

TIMER T0

As seen in figure below, the timer T0 consists of two registers – TH0 and TL0 representing a low and a high byte of one 16-digit binary number.

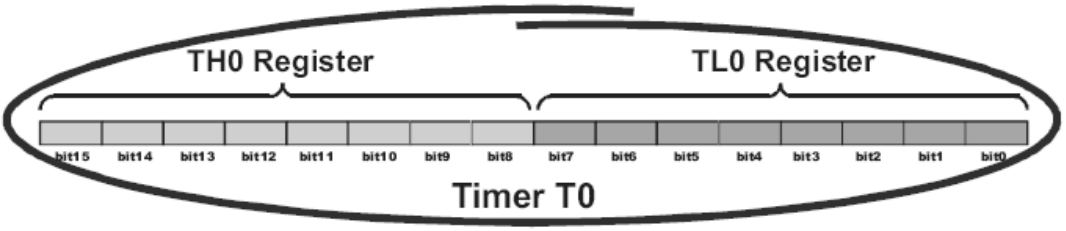

Fig. No. 10: - Timer Register T0

Accordingly, if the content of the timer T0 is equal to 0 (T0=0) then both registers it consists of will contain 0. If the timer contains for example number 1000 (decimal), then the TH0 register (high byte) will contain the number 3, while the TL0 register (low byte) will contain decimal number 232.

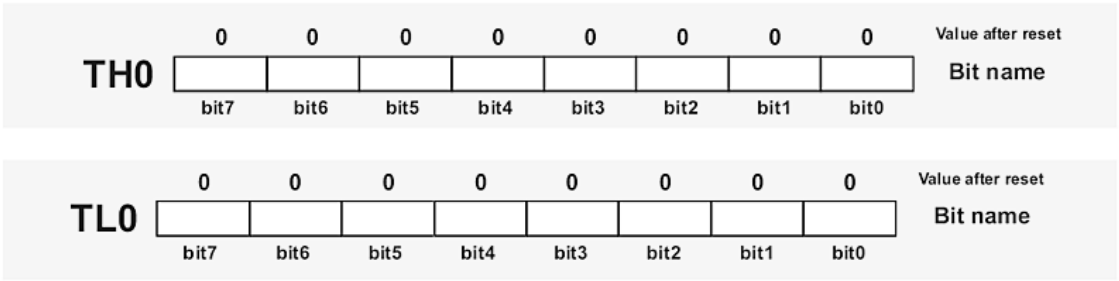

Fig. No. 11: - Timer Register TH0 and TL0

Formula used to calculate values in these two registers is very simple: TH0 × 256 + TL0 = T Matching the previous example it would be as follows: 3 × 256 + 232 = 1000.

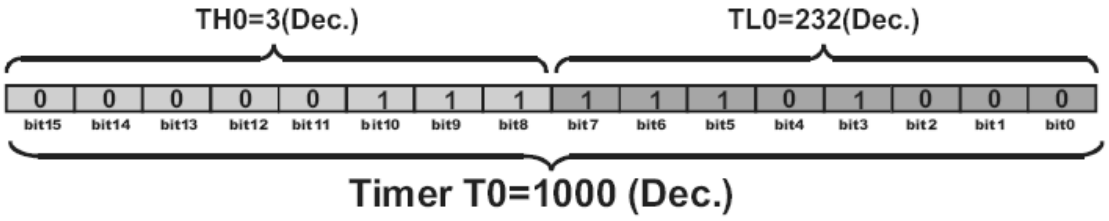

Fig. No. 12: - Value in Timer Register TH0 and TL0

Since the timer T0 is virtually 16-bit register, the largest value it can store is 65 535. In case of exceeding this value, the timer will be automatically cleared and counting starts from 0. This condition is called an overflow. Two registers TMOD and TCON are closely connected to this timer and control its operation.

TMOD REGISTER (TIMER MODE)

The TMOD register selects the operational mode of the timers T0 and T1. As seen in figure below, the low 4 bits (bit0 - bit3) refer to the timer 0, while the high 4 bits (bit4 - bit7) refer to the timer 1. There are 4 operational modes and each of them is described herein.

Fig. No. 13: - TMOD Register

Bits of this register have the following function:

- GATE1 enables and disables Timer 1 by means of a signal brought to the INT1 pin (P3.3):
  - 1 - Timer 1 operates only if the INT1 bit is set.
  - 0 - Timer 1 operates regardless of the logic state of the INT1 bit.
- C/T1 selects pulses to be counted up by the timer/counter 1:
  - 1 - Timer counts pulses brought to the T1 pin (P3.5).
  - 0 - Timer counts pulses from internal oscillator.
- T1M1, T1M0 these two bits select the operational mode of the Timer 1.

T1M1	T1M0	Mode	Description
0	0	0	13-bit timer
0	1	1	16-bit timer
1	0	2	8-bit auto-reload
1	1	3	Split mode

Fig. No. 14: - Operational mode of Timer 1

- GATE0 enables and disables Timer 1 using a signal brought to the INT0 pin (P3.2):
  - 1 - Timer 0 operates only if the INT0 bit is set.
  - 0 - Timer 0 operates regardless of the logic state of the INT0 bit.
- C/T0 selects pulses to be counted up by the timer/counter 0:
  - 1 - Timer counts pulses brought to the T0 pin (P3.4).
  - 0 - Timer counts pulses from internal oscillator.
- T0M1, T0M0 these two bits select the operational mode of the Timer 0.

T0M1	T0M0	Mode	Description
0	0	0	13-bit timer
0	1	1	16-bit timer
1	0	2	8-bit auto-reload
1	1	3	Split mode

Fig. No. 15: - Operational mode of Timer 0

TIMER CONTROL (TCON) REGISTER

TCON register is also one of the registers whose bits are directly in control of timer operation. Only 4 bits of this register are used for this purpose.\

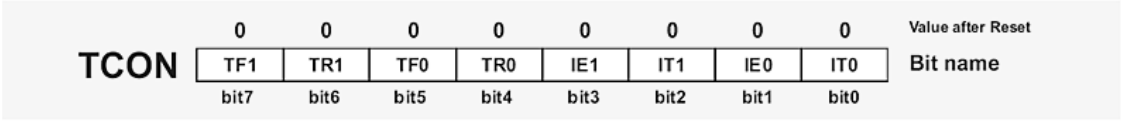

Fig. No. 16: - TCON Register

TF1 bit is automatically set on the Timer 1 overflow.

- TR1 bit enables the Timer 1.
  - 1 - Timer 1 is enabled.
  - 0 - Timer 1 is disabled.
- TF0 bit is automatically set on the Timer 0 overflow.
- TR0 bit enables the timer 0.
  - 1 - Timer 0 is enabled.
  - 0 - Timer 0 is disabled.

Timer 1

Timer 1 is identical to timer 0, except for mode 3 which is a hold-count mode. It means that they have the same function, their operation is controlled by the same registers TMOD and TCON and both of them can operate in one out of 4 different modes.

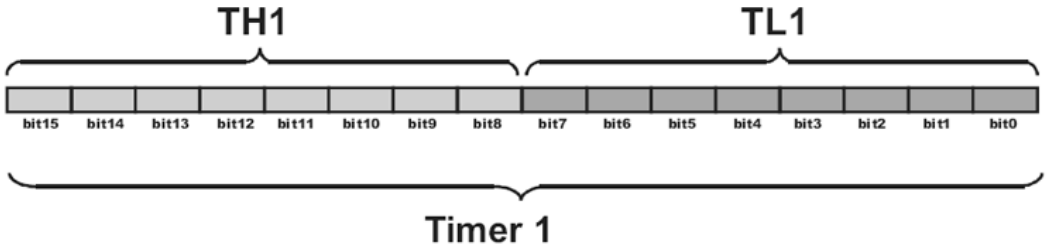

Fig. No. 17: - Timer 1 register

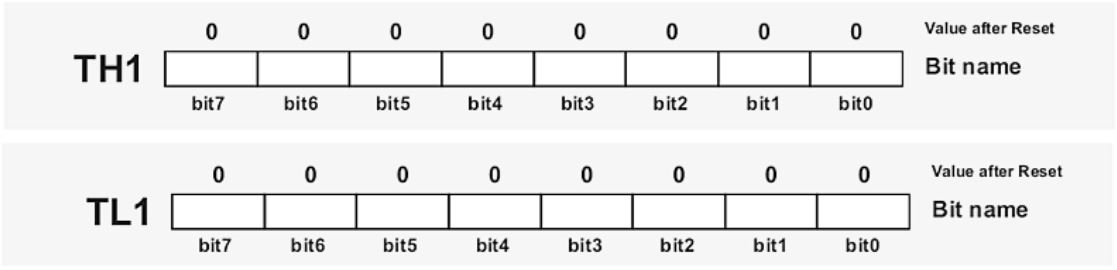

Fig. No. 18: - Timer Register TH1 and TL1

Output: Thus the 8051 Architecture has been studied.

EXP: 8

Aim: Write a program to convert a given Hex number (32H) to Decimal.

Apparatus required: VINYTICS 8051 Microcontroller kit.

Program:

```
HEX TO DECIMAL
; CONVERT PACKED-BCD TO DECIMAL
; R7: PACKED-BCD
; R6: DECIMAL OUTPUT
BCD2DEC:
        MOV A, #32H
        MOV R7,A
        ANL A, #0F0H
        SWAP A
        MOV B, #10
        MUL AB
        MOV R6, A
        MOV A, R7
        ANL A, #0FH
        ADD A, R6
        MOV DPTR, #4062H
        MOVX  @DPTR, A
HERE: SJMP HERE
```

Output:

BEFORE EXECUTION		AFTER EXECUTION	
HEX INPUT	32H	DEC OUTPUT	50

Result:

We have concluded that by number conversion we can convert any format to other that is useful when we have only number in one format.

Discussion:

1) What are Pseudo instructions?
2) Number of the times the instruction sequence below will loop before coming out of loop is

```
MOV AL, 00h
A1: INC AL
JNZ A1
```

3) What is the size of internal RAM memory of the 8051?
4) Explain all number system formats and do conversion by taking 05 as one number in decimal?
5) Why hexadecimal number system is more preferable than other number systems?

EXP: 9

Aim: Write a program to find numbers of even numbers and odd numbers among 10 Numbers (01, 02, 03, 04, 05, 06, 07, 09, 01, 05).

Apparatus required: VINYTICS 8051 Microcontroller kit.

Program:

```
        MOV DPTR, #4000H
        MOV R3, # 0AH
        MOV R1, # 00H       ; for odd number counting
        MOV R2, #00H        ;for even number counting
        CLR C
UP:     MOV B, # 02H
        MOVX  A, @DPTR
        DIV AB
        MOV R0, B
        CJNE R0, #00, ODD
        INC R2
        SJMP SKIP
ODD:  INC R1
SKIP:  INC DPTR
        DJNR R3, UP
        MOV  DPTR, #4062H
        MOV A, R1
        MOVX  @DPTR, A
        INC DPTR
        MOV A, R2
        MOVX @DPTR, A
HERE: SJMP HERE
```

Output:

BEFORE EXECUTION		AFTER EXECUTION	
MEMORY ADDRESS	CONTENT		
4000H	01	4062H	07(ODD)
4001H	02	4063H	03(EVEN)
4002H	03		
4003H	04		
4004H	05		
4005H	06		
4006H	07		
4007H	09		

4008H	01		
4009H	05		

Result:

We have concluded that the even and odd number can be found from a set of array using above algorithm.

Discussion:

1) Intel 8051 Follows Which Architecture?
2) What Is The Difference Between Harvard Architecture And Von Neumann Architecture?
3) Explain DJNZ, CJNE, and MOVX?
4) List out the features of 8051 Microcontroller?
5) What are the types of Interrupts in 8051?

EXP: 10

Aim: Write a program to find Largest and Smallest Numbers among 10 Numbers (01, 02, 03, 04, 05, 06, 07, 09, 01, 05).

Apparatus required: VINYTICS 8051 Microcontroller kit.

Program:

```
MOV R3, # 0AH
MOV DPTR, #4000H
MOVX A, @DPTR
MOV R1, A
JMP: INC DPTR
MOVX A, @DPTR
CLR C
MOV R2, A
SUBB A, R1
JC SKIP                (JC – LARGEST/ JNC - SMALLEST)
MOV A, R2
MOV R1, A
SKIP: DJNZ R3, JMP
MOV DPTR, #4062H
MOV A, R1
MOVX @DPTR, A
HERE: SJMP HERE
```

Output:

BEFORE EXECUTION		AFTER EXECUTION	
MEMORY ADDRESS	CONTENT		
M3000	01	Largest No.	09
M3001	02	Smallest No.	01
M3002	03		
M3003	04		
M3004	05		
M3005	06		
M3006	07		
M3007	09		
M3008	01		
M3009	05		

Result:

We have concluded that the largest and smallest number can be found from a set of array using above algorithm.

Discussion:

1) Are all the bits of flag register used in 8051?
2) What are the four distinct types of memory in 8051?
3) Give example of bit address and byte address?
4) Explain DPTR, SUBB?
5) What location code memory space and data memory space begins?

EXP: 11.1

Aim: Write a program to generate a delay of 20 ms with Loop.

Apparatus required: NV5001 Microcontroller Development board with Programmer kit.

Software required: Keil µvision 5, Proload 8051 flash program.

Component required: LED

Theory: The factor affects the delay is :

(1) The number of machine cycle and the number of clock periods per machine cycle vary among different versions of the microcontroller, So time delay depends on the version or type of microcontroller.

(2) It depends on the crystal frequency connected to the X1-X2 input pins.

(3) Compiler choice: The third factor that affects the time delay is the compiler used to compile the c program.When program in assembly language, we an control the exact instruction and their sequence used in the delay subroutine. In the case of c program, it's the C compiler that converts the C statements and functions to assembly language instruction. As a result, different compiler produce different code. In other words, if we compile a given 8051 C program with different compiler, each compiler produce different hex code.

So when we write time delay for C, we must use the oscilloscope to measure the exact duration.

Program:

```
#include <reg51.h>                  //Define 8051 Registers
sbit LED =P2^0;                    //Set the bit in P2^0
void Delay (unsigned int a);        //Delay function
//-------------------------
//      Main Program
//-------------------------
void main()
{
     LED=0
     while(1)                      //Loop forever
    {
       LED = 1;                   //Set the bit0 in port1 to high
      Delay (200);                //Delay time for 20ms
     LED = 0;                    //Set the bit0 in port1 to low
     Delay(200);                 //Delay time for 20ms
    }
}
```

```
///---------------------------------------
// DELAY at 11.0592MHz crystal.
// Calling the routine takes about 22us, and then
// each count takes another 1.02ms.//---------------------------------------

void Delay(unsigned int count)
{      unsigned int i;
      while(count)
{
      i = 115;
     while(i>0) i--;
     count--;
}
}
```

Output: Here we see that how we can apply delay using loop and on/off an LED (P2.).

Result: From results seen we can say loops are used to make delays and can be used to give any delay. In the similar manner we can make disco lights, chaser lights, running lights etc.

Discussion

1) How many bytes is the bidirectional input/output port.
2) What is the principle behind interfacing LEDs with 8051 microcontroller?
3) How much current LEDs need approximately to flow through them in order to glow at maximum intensity.
4) Explain: LEDs run on which logic.
5) How is delay provided to process LED blinking.

EXP: 11.1

Aim: Write a program to generate a delay of 10 ms with Timer.

Apparatus required: NV5001 Microcontroller Development board with Programmer kit.

Software required: Keil µvision 5, Proload 8051 flash program.

Component required: LED

Theory:

While designing delay programs in 8051, calculating the initial value that has to be loaded into TH and TL registers forms a very important thing.

Assume the processor is clocked by 11.059 MHz crystal.

That means, the timer clock input will be 11.059 /12 = 92158 MHz

That means, the time taken for the timer to make one increment = 1/92158MHz = 1.085 micro seconds

For a time delay of "X" uS the timer has to make "X" increments.

2^16 = 65536 is the maximum number of counts possible for a 16 bit timer.

Let TH be the value that has to be loaded to TH register and TL be the value that has to be loaded to TL register.

Then, THTL = Hexadecimal equivalent of (65536-X) where (65536-X) is considered in decimal.

DELAY USING TIMER

For delay of 10 mili seconds

N= 10ms/1.085 µS =9216

M-N = 65536-9216=56320 SO IN HEX = DC00

Program:

```
#include <reg51.h>                  //Define 8051 Registers
sbit LED =P2^0;                    //Set the bit in P2^0
void Delay (unsigned int a);        //Delay function
//-------------------------
//      Main Program
//-------------------------
```

```
void main()
{
      LED=0
      while(1)                    //Loop forever
     {
        LED = 1;                 //Set the bit0 in port1 to high
       Delay (200);               //Delay time for 10ms
      LED = 0;                  //Set the bit0 in port1 to low
      Delay(200);               //Delay time for 10ms
     }
}
///---------------------------------------
// DELAY at 11.0592MHz crystal.

void Delay(unsigned int count)
{
TMOD=0X01
TH0=0XDC
TL0=0X00
TR=1
While(TF0=0)
{
TR0=0;
TF0=0;
}
```

Output: Here we see that how we can apply internal timers to provide delays and put an LED ON/OFF (P2.).

Result: From results seen we can say there can be other approach of providing delays i.e. internal timers (TMOD) which can be used to make delays and can be used to give any delay. In the similar manner we can make disco lights, chaser lights, running lights etc.

Discussion:

1) What is the function of the TMOD register?
2) What is the maximum delay that can be generated with the crystal frequency of 22MHz?
3) In the instruction “MOV TH1, #-3”, what is the value that is being loaded in the TH1 register?
4) Find out the roll over value for the timer in Mode 0, Mode 1 and Mode 2?
5) If Timer 0 is to be used as a counter, then at what particular pin, clock pulse needs to be applied?

EXP: 11.3

Aim: Write a program to generate a square wave on output pin using timer on CRO.

Apparatus required: NV5001 Microcontroller Development board with Programmer kit, CRO.

Software required: Keil µvision 5, Proload 8051 flash program.

Component required: LED

Delay calculation:

Delay of 1.085µs:

N=1.085 µs/ 1.085 µs=1cycle

65536-1=65535 or FFFFH in Hex

Program:

```
#include <reg51.h>                    //Define 8051 Registers
sbit LED =P2^0;                      //Set the bit in P2^0
void Delay (unsigned int a);          //Delay function
//-------------------------
//      Main Program
//-------------------------
void main()
{
     LED=0
     while(1)                        //Loop forever
    {
       LED = 1;                  //Set the bit0 in port1 to high
      Delay (200);               //Delay time for 20ms
     LED = 0;                   //Set the bit0 in port1 to low
     Delay(200);                //Delay time for 20ms
    }
}
///--------------------------------------

// DELAY at 11.0592MHz crystal.(Delay time:1.085µs)


void Delay(unsigned int count)
{
TMOD=0X01
TH0=0XFF
```

```
TL0=0XFF
TR=1
While(TF0=0)
{
TR0=0;
TF0=0;
}
```

Output: Here we can see that how we can apply internal timers to provide delays and generate square wave at CRO.

Result:

From results seen we can infer that there can be other approaches of providing delays i.e. internal timers (TMOD) which can be used to make delays and can be used to generate a signal which can be plotted on CRO.

Discussion:

1) Explain format of TMOD register?
2) Explain all modes of TMOD register?
3) Explain format of TCON register?
4) What is general formula of timer? Calculate HEX value for time delay 5ms?

EXP: 12.1

Aim: Write a program to interface Seven Segment Display with 8051.

Apparatus required: NV5001 Microcontroller Development board with Programmer kit.

Software required: Keil µvision 5, Proteus 8 professional, 8051 flash programmer (Proload).

Component Required:

COMPONENT NAME	TYPE	QUANTITY
Microcontroller	AT89S52	1
Crystal Oscillator	11.0592 MHz	1
Seven Segment	Common anode	1
Resistor	10 KΩ	1
	1 KΩ	8
	470Ω	1
	10 KΩ(POT)	1
Capacitor	10µF	1
	33 pf	2
Push Button		1

Theory:

Seven segment LED display is very popular and it can display digits from 0 to 9 and quite a few characters like A, b, C, H, E, e, F, n, o, t, u, y, etc. Knowledge about how to interface a seven segment display to a micro controller is very essential in designing embedded systems. A seven segment display consists of seven LEDs arranged in the form of a squarish **'8'** slightly inclined to the right and a single LED as the dot character. Different characters can be displayed by selectively glowing, the required LED segments. Seven segment displays are of two types, **common cathode and common anode.** In common cathode type , the cathode 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 & h (or dot) . In common anode type, the anode of all LEDs, are tied together as a single terminal and cathodes are left alone as individual pins. The pin layout scheme and picture of a typical 7 segment LED display is shown in the image below.

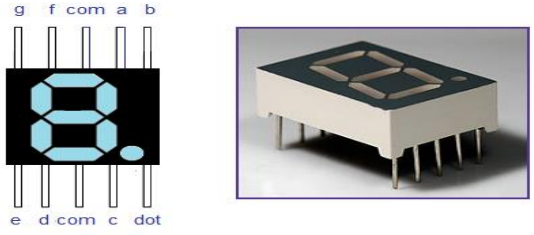

Fig. No. 12(a).1: - Seven segment LED display

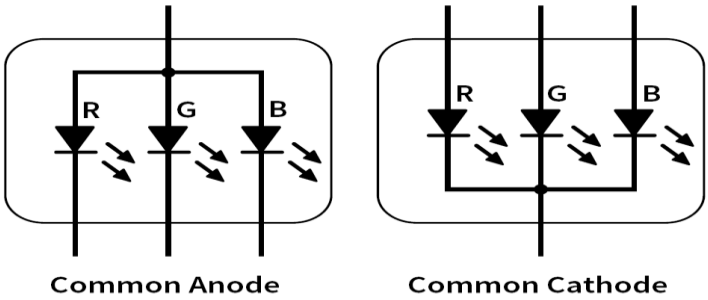

Fig. No. 12(a).2: - Common Cathode and Common Anode

Digit drive pattern:-

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.

For Common anode:

Digit to Display	h g f e d c b a	Hex code
0	1 1 0 0 0 0 0 0	0xC0
1	1 1 1 1 1 0 0 1	0xF9
2	1 0 1 0 0 1 0 0	0xA4
3	1 0 1 1 0 0 0 0	0xB0
4	1 0 0 1 1 0 0 1	0x99
5	1 0 0 1 0 0 1 0	0x92
6	1 0 0 0 0 0 1 0	0x82
7	1 1 1 1 1 0 0 0	0xF8
8	1 0 0 0 0 0 0 0	0x80
9	1 0 0 1 0 0 0 0	0x90

For Common cathode:

Digit to Display	h g f e d c b a	Hex code
0	0 0 1 1 1 1 1 1	0xC0
1	0 0 0 0 0 1 1 0	0xF9
2	0 1 0 1 1 0 1 1	0xA4
3	0 1 0 0 1 1 1 1	0xB0
4	0 1 1 0 0 1 1 0	0x99
5	0 1 1 0 1 1 0 1	0x92
6	0 1 1 1 1 1 0 1	0x82
7	0 0 0 0 0 1 1 1	0xF8
8	0 1 1 1 1 1 1 1	0x80
9	0 1 1 0 0 1 1 1	0x90

Interfacing seven segment display to 8051:-

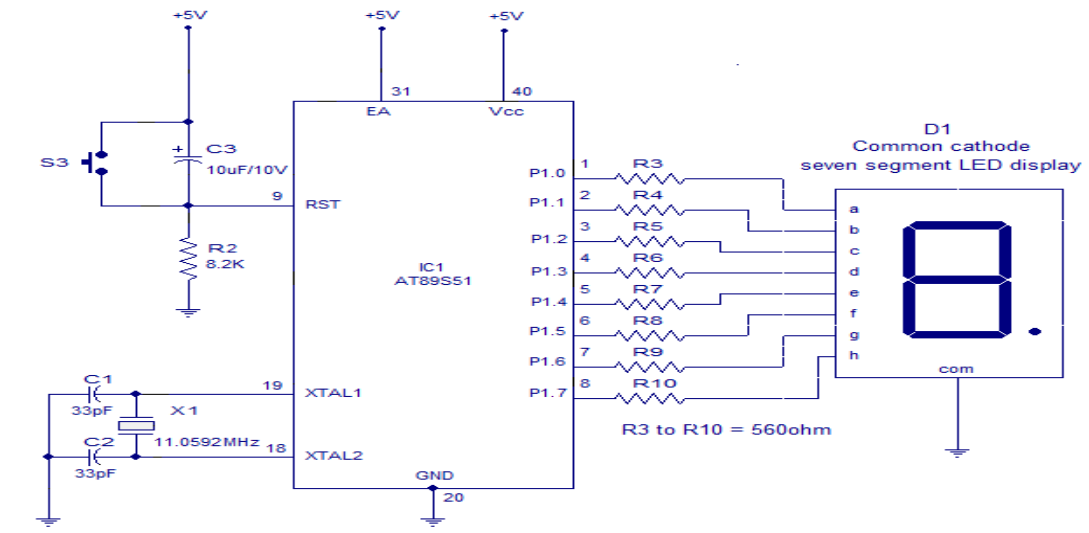

Fig. No. 12(a).3: - Interfacing seven segment display to 8051

The circuit diagram shown above is of an AT89S51 microcontroller based 0 to 9 counter which has a 7 segment LED display interfaced to it in order to display the count. This simple circuit illustrates two things: how to setup simple 0 to 9 up counter using 8051 and more importantly how to interface a seven segment LED display to 8051 in order to display a particular result. The common cathode seven segment display D1 is connected to the Port 1 of the microcontroller (AT89S51) as shown in the circuit diagram. R3 to R10 are current limiting resistors. S3 is the reset switch and R2 and C3 forms a de-bouncing circuitry. C1, C2 and X1 are related to the clock circuit.

Proteus Circuit:

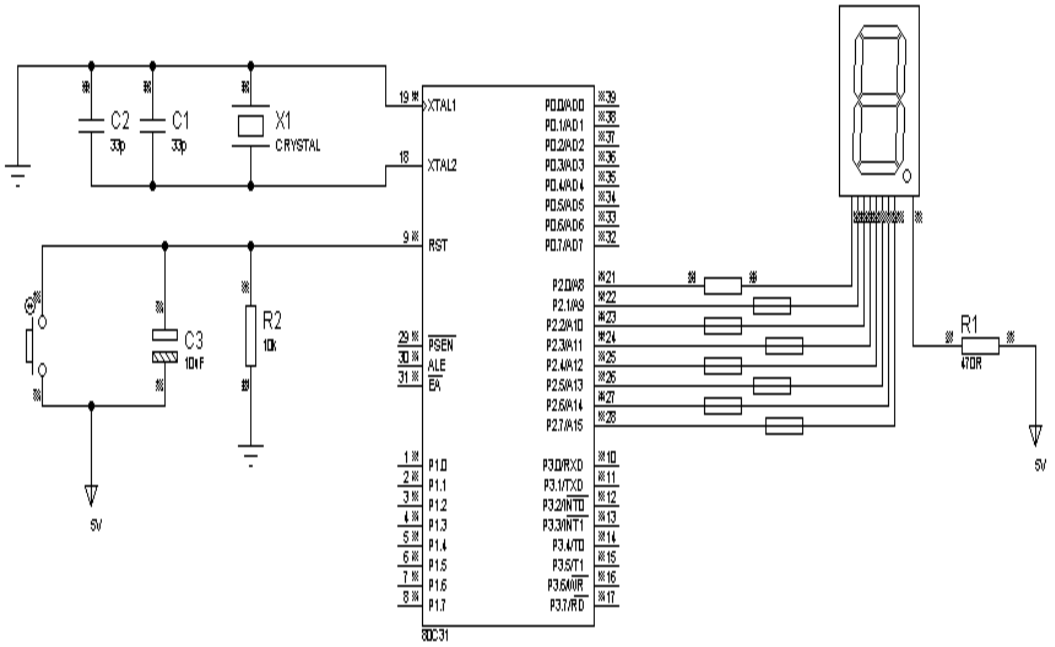

Fig. No. 12(a).3: - Implementation diagram (Proteus) seven segment display to 8051

Keil Program:

```
#include<reg51.h>

void msdelay(unsigned int time)  // Function for creating delay in milliseconds.
{
   unsigned i,j ;
   for(i=0;i<time;i++)
   for(j=0;j<1275;j++);
}

void main()
{
   unsigned char no_code[]={0xC0,0xF9,0xA4,0xB0,0x99,0x92,0x82,0xF8,0x80,0x90};
//Array for hex values (0-9) for common anode 7 segment
   int k;
   while(1)
   {
      for(k=0;k<10;k++)
      {
       P0=no_code[k];
       msdelay(100);
      }
   }
}
```

Output: We saw that seven-segment is interfaced with 8051 and “0 to 9” is continuously displayed on it with a small delay.

Result: We can conclude that we can easily interface display circuits with 8051 and use then at different applications like clocks, indicators etc.

Discussion:

1) Explain seven-segment with diagram?
2) What is common cathode and anode mode of IC’s?
3) With help of KMAPs solve all input equations for seven-segment?
4) Explain traffic light application that uses a seven segment display as counter?

EXP: 12.2

Aim: Write a program to interface LCD with 8051.

Apparatus required: NV5001 Microcontroller Development board with Programmer kit.

Software required: Keil µvision 5, Proteus 8 professional, 8051 flash programmer (Proload).

Component Required:

COMPONENT NAME	TYPE	QUANTITY
Microcontroller	AT89S52	1
Crystal Oscillator	11.0592 MHz	1
LCD	16 x2	1
Resistor	10 KΩ	1
	10 KΩ(POT)	1
Capacitor	10µF	1
	33 pf	2
Push Button		4

Theory:

16×2 LCD module is a very common type of LCD module that is used in 8051 based embedded projects. It consists of 16 rows and 2 columns of 5×7 or 5×8 LCD dot matrices. The module we are talking about here is JHD162A (type number) which is a very popular one. It is available in a 16 pin package with back light, contrast adjustment function and each dot matrix has 5×8 dot resolution. The pin numbers, their name and corresponding functions are shown in the table below.

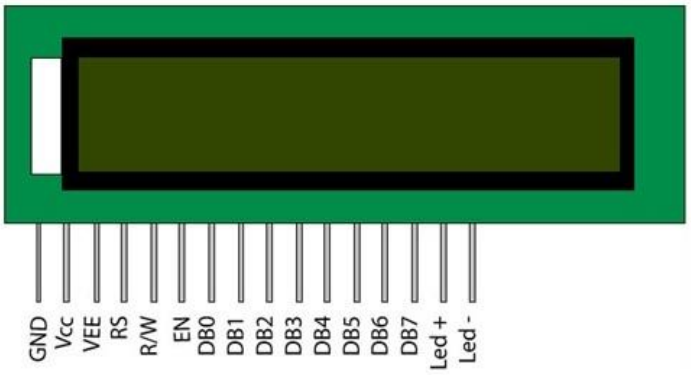

Fig. No. 12(b).1: - Liquid Crystal Display

Category	Pin NO.	Pin Name	Function
Power Pins	1	VSS	Ground Pin, connected to Ground
	2	VDD or Vcc	Voltage Pin +5V
Contrast Pin	3	V0 or VEE	Contrast Setting, connected to Vcc thorough a variable resistor.

Control Pins	4	RS	Register Select Pin, RS=0 Command mode, RS=1 Data mode
	5	RW	Read/ Write pin, RW=0 Write mode, RW=1 Read mode
	6	E	Enable, a high to low pulse need to enable the LCD
Data Pins	7-14	D0-D7	Data Pins, Stores the Data to be displayed on LCD or the command instructions
Backlight Pins	15	LED+ or A	To power the Backlight +5V
	16	LED- or K	Backlight Ground

16×2 LCD module commands

16×2 LCD module has a set of preset command instructions. Each command will make the module to do a particular task. The commonly used commands and their function are given in the table below.

Command	Function
0F	LCD ON, Cursor ON, Cursor blinking ON
01	Clear screen
02	Return home
04	Decrement cursor
06	Increment cursor
0E	Display ON ,Cursor blinking OFF
80	Force cursor to the beginning of 1stline
C0	Force cursor to the beginning of 2ndline
38	Use 2 lines and 5×7 matrix
83	Cursor line 1 position 3
3C	Activate second line
08	Display OFF, Cursor OFF
C1	Jump to second line, position1
OC	Display ON, Cursor OFF
C1	Jump to second line, position1
C2	Jump to second line, position2

LCD initialization:-

The steps that are been taken for initializing the LCD display are given below:

- Send 38H to the 8 bit data line for initialization
- Send 0FH for making LCD ON, cursor ON and cursor blinking ON.
- Send 06H for incrementing cursor position.
- Send 01H for clearing the display and return the cursor.
- Sending data to the LCD.

The steps that are been taken for sending data to the LCD module are given below.

- Make R/W low.
- Make RS=0 if data byte is a command and make RS=1 if the data byte is a data to be displayed.
- Place data byte on the data register.
- Pulse E from high to low.
- Repeat above steps for sending another data.

Circuit Diagram

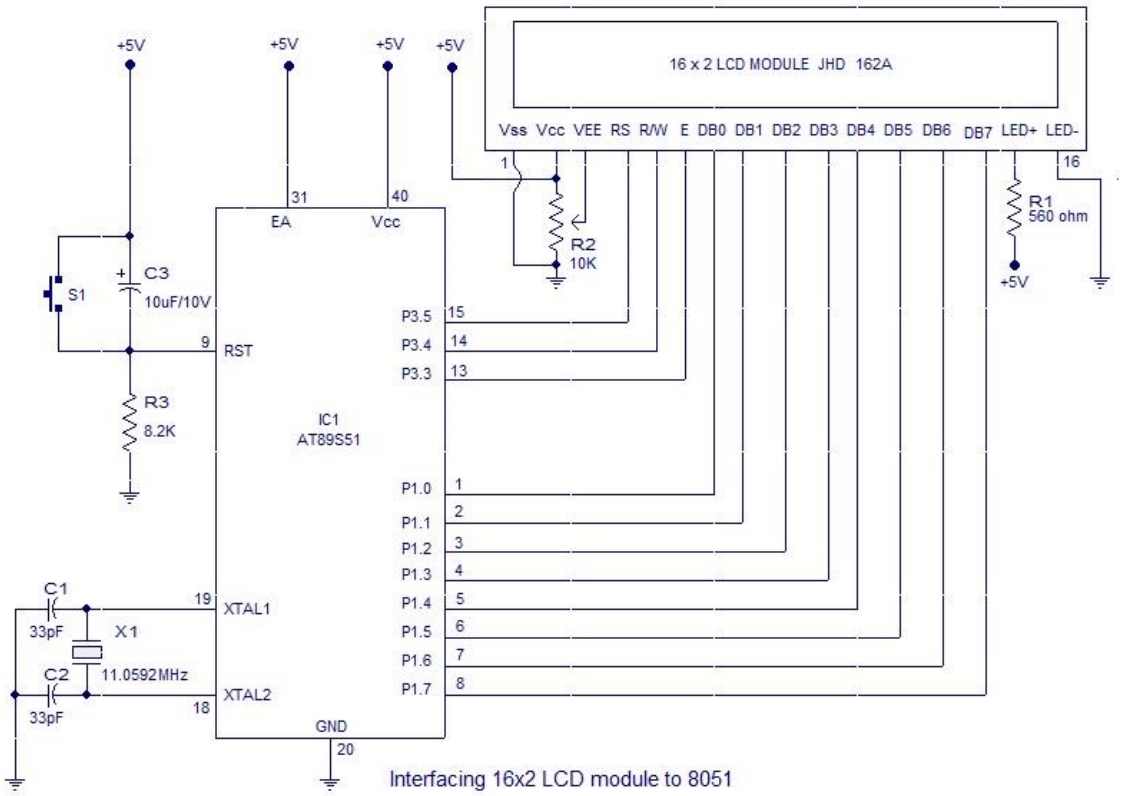

Fig. No. 12(b).2: Interfacing LCD with 8051

Interfacing 16×2 LCD module to 8051

The circuit diagram given above shows interfacing of a 16×2 LCD modules with AT89S1 microcontroller. Capacitor C3, resistor R3 and push button switch S1 forms the reset circuitry. Ceramic capacitors C1, C2 and crystal X1 is related to the clock circuitry which produces the system clock frequency. P1.0 to P1.7 pins of the microcontroller is connected to the DB0 to DB7 pins of the module respectively and through this route the data goes to the LCD module. P3.3, P3.4 and P3.5 are connected to the E, R/W, RS pins of the microcontroller and through this route the control signals are transferred to the LCD module. Resistor R1 limits the current through the back light LED and so do the back light intensity. POT R2 is used for adjusting the contrast of the display.

Proteus Circuit:

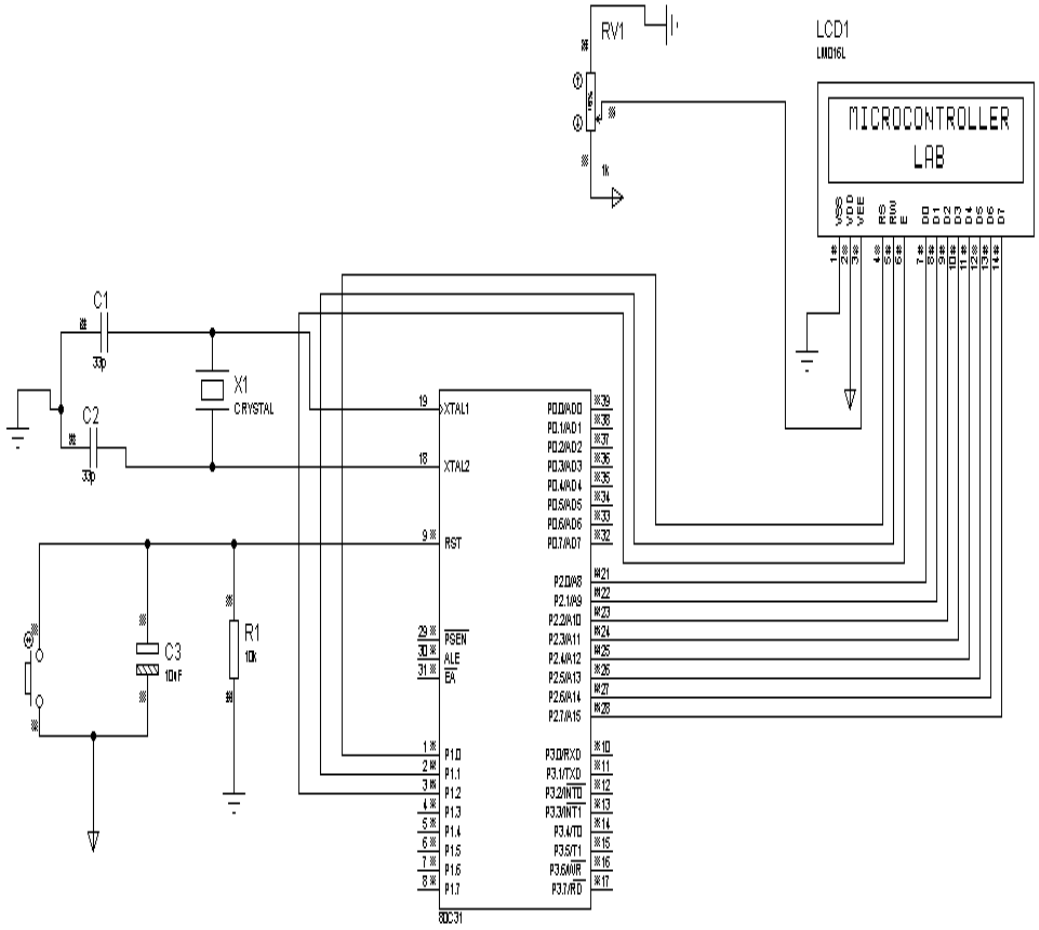

Fig. No. 12(b).3: - Implementation diagram (Proteus) LCD to 8051

Keil Program:

```
#include<reg51.h>
#define lcd P2

sbit rs=P1^0;
sbit rw=P1^1;
sbit e=P1^2;

void delay (int);
void cmd (char);
void display (char);
void string (char *);
void init (void);

void delay (int d)
{
        unsigned char i;
        for(;d>0;d--)
```

```
        {
                for(i=250;i>0;i--);
                for(i=248;i>0;i--);
        }
}
void cmd (char c)
{
        lcd=c;
        rs=0;
        rw=0;
        e=1;
        delay(5);
        e=0;
}
void display (char c)
{
        lcd=c;
        rs=1;
        rw=0;
        e=1;
        delay(5);
        e=0;
}

void string (char *p)
{
        while(*p)
        {
                display(*p++);
        }
}
void init (void)
{
        cmd(0x38);
        cmd(0x0c);
        cmd(0x01);
        cmd(0x80);
}
void main()
{
        while(1)
        {
        init();
        cmd(0x83);
        string("WELCOMES SKIT ");
        cmd(0xc4);
        string("    JAGATPURA.  ");
        delay(400);
        cmd(0x01);
```

```
cmd(0x81);
string("MICROCONTROLLER");
cmd(0xc5);
string(" LAB ");
delay(400);
}
```

Output: The message given in the program is displayed as shown in figure of LCD below –

Fig. No. 12(b).4: - Output Display

Result: From the result, we can conclude that the text can be displayed on the LCD and shifting of this text is possible in both left and right directions.

Discussion:

1) Explain 16*2 LCD module commands.
2) How we initialize LCD?
3) Give module description of 16*2 LCD.
4) What is the significance of interfacing a LCD with microcontroller?
5) Is it possible to move the text which is displaying on LCD on both sides?

EXP: 13

Aim: Write a program for Traffic light Control using 8051.

Apparatus required: NV5001 Microcontroller Development board with Programmer kit.

Software required: Keil µvision 5, Proteus 8 professional, 8051 flash programmer (Proload).

Component Required:

COMPONENT NAME	TYPE	QUANTITY
Microcontroller	AT89S52	1
Crystal Oscillator	11.0592 MHz	1
LED	RED	04
	YELLOW	04
	GREEN	04
	BLUE	04
Resistor	10 KΩ	2
Capacitor	10µF	01
	33 pf	02
Push Button		01

Theory:

Traffic lights were first invented in the year 1868 at London's House of Commons where traffic light signals were placed at intersections of George and Bridge Street. Later the traffic lights were developed in the year 1914 by an American Traffic Signal Company, which fixed green and red lights at corners of the 105th street and Euclid Avenue in Cleveland, Ohio. During this period traffic lights were controlled either by timing or by switching manually.

Traffic lights are also named as stoplights, road traffic lamps, traffic signals, stop-and-go lights which are signaling devices placed at road crossings, everyday pedestrian crossings and other locations to control competing flows of traffic. Traffic lights have been fixed all over the world in many cities. Traffic light control assigns a right way to the road users by using lights in normal colors (red – amber/yellow – green). Traffic light control system uses a worldwide color code (a specific color order to enable color recognition for those who are color blind).

Typically traffic lights consist of three types of colored lights such as red, orange and green. In a typical cycle, turning on of a green light allows traffic to continue in the way indicated. Similarly, lighting of the amber/orange light for a short time of transition represents a signal to prepare to stop, and the illumination of the red signal disallows any traffic from going on.

A control system is necessary to control these lights in a specific manner. This traffic light control system can be achieved by using a microcontroller to make a simple and low-cost system.

Fig. No.13.1: - Illustration of Traffic Light System

Traffic Light Controller using Microcontroller

The main objective of this traffic light controller is to provide sophisticated control and coordination to confirm that traffic moves as smoothly and safely as possible. This experiment makes use of LED lights for indication purpose and a microcontroller is used for auto changing of signal at specified range of time interval. LED lights get automatically turn ON and OFF by making corresponding port pin of the microcontroller "HIGH".

Working of the Traffic Light Controller

In the below circuit diagram of traffic light controller four LEDs are used for the purpose of traffic light control. An 8051 Microcontroller is the brain of this whole experiment and is used to initiate the traffic signal at the intersections on road.

The LEDs get automatically switched ON and OFF by making the corresponding port pins of the microcontroller high, based on the 8051 microcontroller and its programming is done by using KEIL software. At a particular period of time, only the green light holds ON and the other lights remains OFF, and after sometime, the changeover traffic light control from green to red takes place by making the succeeding change for glowing of yellow LED. This process continues as a cycle and the timing for changing the LEDs.

Proteus circuit:

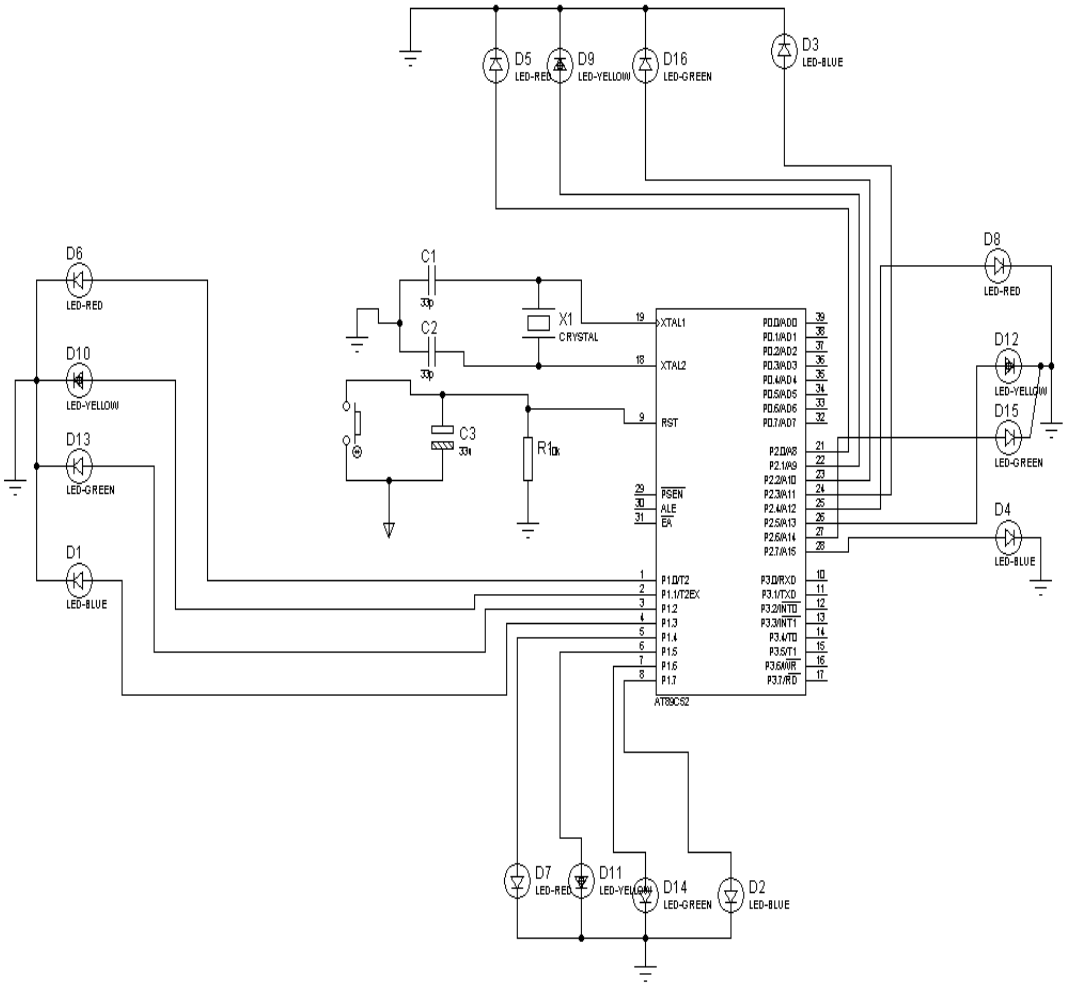

Fig. No.13.2: - Implementation diagram (Proteus) traffic light

Keil Code:

```
#include<reg51.h>
sbit R1=P1^4;
sbit Y1=P1^5;
sbit G1=P1^6;
sbit B1=P1^7;
sbit R2=P2^4;
sbit Y2=P2^5;
sbit G2=P2^6;
sbit B2=P2^7;
sbit R3=P2^0;
sbit Y3=P2^1;
sbit G3=P2^2;
sbit  B3=P2^3;
sbit R4=P1^0;
sbit Y4=P1^1;
sbit G4=P1^2;
sbit B4=P1^3;
```

```
void msdelay(unsigned int t)
{
int i,j;
int k=100*t;
for(i=0;i<k;i++)
for(j=0;j<1275;j++);
}

void clear()
{
R1=0;
R2=0;
R3=0;
R4=0;
Y1=0;
Y2=0;
Y3=0;
Y4=0;
G1=0;
G2=0;
G3=0;
G4=0;
B1=0;
B2=0;
B3=0;
B4=0;
}


void phase1()
{

G3=1;
B3=1;
R1=1;
R4=1;
R2=1;
msdelay(7);
Y3=1;
G3=0;
B3=0;
msdelay(3);
}

void phase2()
{
G2=1;
B2=1;
R1=1;
R4=1;
R3=1;
```

```
msdelay(7);
Y2=1;
G2=0;
B2=0;
msdelay(3);
}

void phase3()
{
G1=1;
B1=1;
R4=1;
R3=1;
R2=1;
msdelay(7);
Y1=1;
G1=0;
B1=0;
msdelay(3);
}

void phase4()
{
G4=1;
B4=1;
R3=1;
R2=1;
R1=1;
msdelay(7);
Y4=1;
G4=0;
B4=0;
msdelay(3);
}


void main()
{

while(1)
  {
clear();
 phase1();
 clear();
 phase2();
 clear();
 phase3();
 clear();
 phase4();
 clear();
  }

}
```

Output: Traffic Signal control is carried out for four lanes in which green and red lights turns ON for 10 seconds while yellow light turns ON for 2 seconds.

Result: We have concluded that by above method, how we can implement real life problems like traffic light controller using 8051 microcontroller.

Discussion:

1) Explain Traffic light system running worldwide?
2) Explain how loops are generated in C language?
3) Implement the above code with help of seven segment?
4) Explain phase, ms-delay and clear functions used in coding?

EXP: 14

Aim: Design and Implement logic for Elevator Control using 8051.

Apparatus required: NV5001 Microcontroller Development board with Programmer kit.

Software required: Keil µvision 5, Proteus 8 professional, 8051 flash programmer (Proload).

Component Required:

COMPONENT NAME	TYPE	QUANTITY
Microcontroller	AT89S52	1
Crystal Oscillator	11.0592 MHz	1
LED	GREEN	08
	YELLOW	04
Resistor	10 KΩ	2
Capacitor	10µF	01
	33 pf	02
Push Button		04

Theory: Elevator control system

User specifies/select the floor on which he wants to move to. Push buttons are used to select the floor. There are total four dummy floors in this lift control system. After selecting the floor lift/elevator starts to move to the user selected floor. Once the elevator reaches to the users' desired floor, to select the floor/level there are four buttons for each floor/level. LEDs are attached with the system which shows the status of the lift and its current presence on the floor/level. A led bar is connected to the system which depicts the floor through which the elevator is crossing or is present.

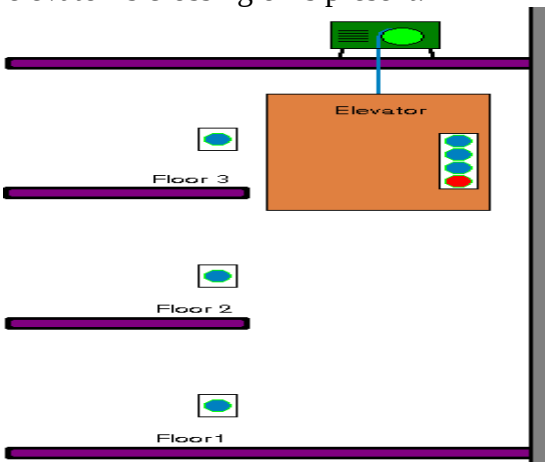

Fig. No.14.1: - Illustration of Elevator Control System

Actual Elevator Control System: In the figure above, there are four buttons: buttons 1, 2, 3 to go to the respective floor, and a stop button to stop the elevator. On each floor, there is

a button to call the elevator. The elevator is moved up and down by a motor (DC or STEEPER MOTOR) that is on the roof of the building. The motor will rotate either clockwise or counterclockwise based on how you want the elevator to move. There is a pulley attached to the motor, and as it rotates, the pulley either rewinds or releases some of the string that is also attached to the elevator.

To keep track of where the elevator is: There is a sensor (photocell/ IR etc.) on each floor. When the elevator gets to one of the floors, the sensor (photocell/ IR etc.) that is on that floor will receive less light. This way the microcontroller knows where the elevator currently is, and uses the display to show the current floor.

IR Sensor (Infrared sensor) module: In IR sensor module 2 parts are present, first one is IR LED and second one is photodiode. It is known as IR pair or photo coupler. The principle of IR sensor is based on LED emission. Hence, IR LED emits IR radiation and photodiode sense that radiation .the resistance of photodiode vary due to IR radiation falling on photodiode. When light is fall onto the photodiode voltage drop generate across the voltage comparator (LM358). Due to this voltage drop IR sensor sense the objects.

Motor driver IC (L 293D): It is 16 pin IC which use to control the rotation of DC motor. In this IC there are two enable pins i.e. pin1 and pin9. This is necessary to high pin1 and pin9 for driving the motor. DC gear/ Steeper motor can be used for rotation in both directions i.e. clockwise rotation and anticlockwise rotation respectively.

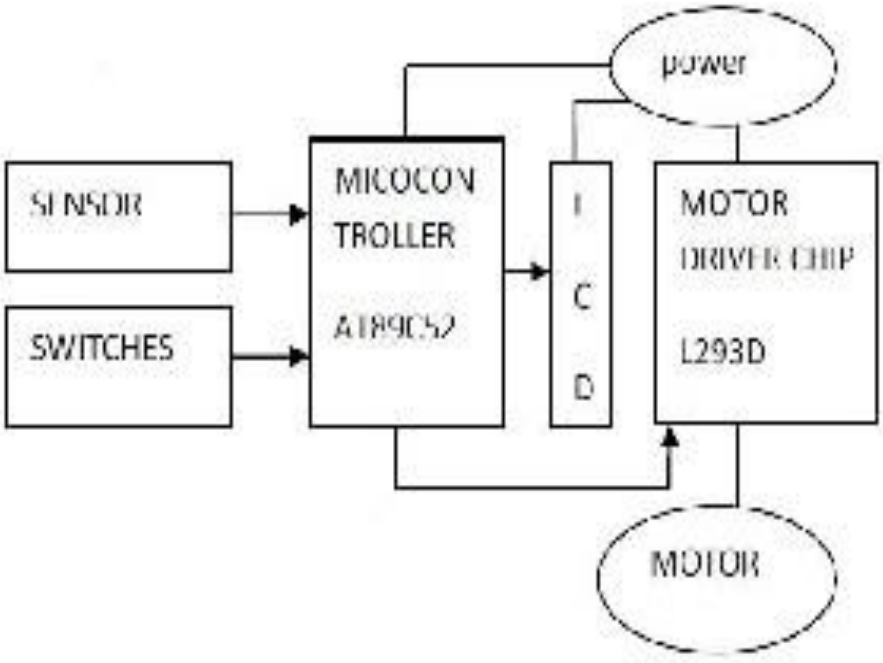

Fig. No.14.2: - Elevator Control System

Designing - Delay Calculation:

While designing delay programs in 8051, calculating the initial value that has to be loaded into loop

Assume the processor is clocked by 11.059 MHz crystal.

That means, the timer clock input will be 11.059 /12 = 92158 MHz

That means, the time taken for the timer to make one increment = 1/92158MHz = 1.085 micro seconds

For a time delay of "X" uS the timer has to make "X" increments.

2^16 = 65536 is the maximum number of counts possible for a 16 bit timer.

Then, Count Value = Hexadecimal equivalent of (65536-X) where (65536-X) is considered in decimal.

Delay Using Timer:

For delay of 50 mili seconds

= 50 mS/1.085uS = 46083

= M-N = 65536 – 46083 = 19453

The value of j if i=50 is 390 for this program.

Formula for Calculating the Crystal load capacitors:

The following formula may be used to calculate a parallel resonant crystal's external load capacitors:

CL = ((CX1 x CX2) / (CX1 + CX2)) + Cstray

where:

CL = the crystal load capacitance
Cstray = the stray capacitance in the oscillator circuit, which will normally be in the 2pF to 5pF range.

Assuming that CX1=CX2 then the equation becomes:

CL = ((CX1 x CX1) / (2 x CX1)) + Cstray
CL = (CX1 / 2) + Cstray
Rearranging the equation, we can find the external load capacitor value:

CX1 = 2(CL - Cstray)

For example, if the crystal load capacitance is 15pF, and assuming Cstray=2pF, then:

CX1 = CX2 = 2(15pF - 2pF) = 26pF

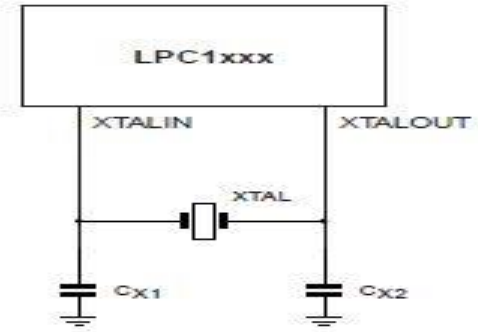

Fig. No.14.3: - Crystal load capacitors

Proteus Circuit:

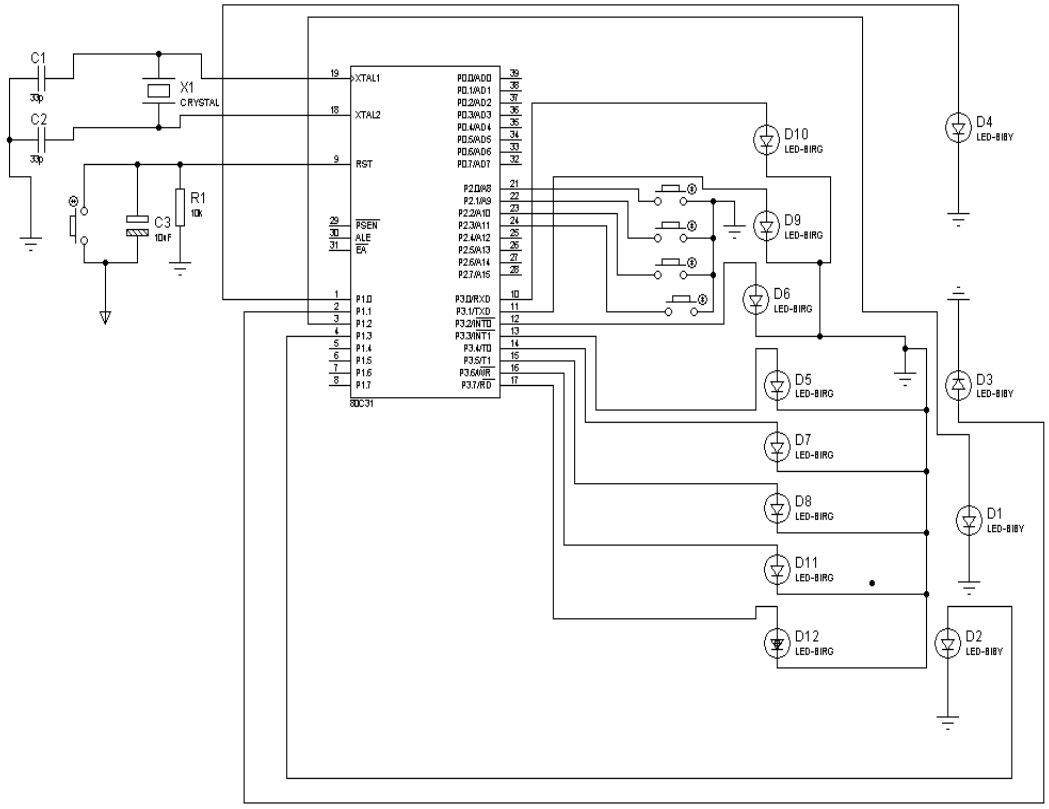

Fig. No.14.4: - Implementation diagram (Proteus) of elevator control system

Keil Code:

```
# include<reg52.h>

void delay (unsigned int ms) ;
void floor1();
void floor2();
void floor3();
void floorground();

sbit led0 = P1^0; //Defining LED PIN
```

```
sbit led1 = P1^1;
sbit led2 = P1^2;
sbit led3 = P1^3;
sbit sw0 = P2^0; //Defining Switch PIN
sbit sw1 = P2^1;
sbit sw2 = P2^2;
sbit sw3 = P2^3;

sbit ledb1=P3^0;
sbit ledb2=P3^1;
sbit ledb3=P3^2;
sbit ledb4=P3^3;
sbit ledb5=P3^4;
sbit ledb6=P3^5;
sbit ledb7=P3^6;
sbit ledb8=P3^7;

void main (void)
{
P1=0X00;
P3=0X00;
P2=0XFF;
        ledb1=1;
        while(1)
        {

        if(sw0==0)
        {
                                floorground();
        }
        if(sw1==0)
        {

                floor1();
        }
        if(sw2==0)
        {

                floor2();
        }
                if(sw3==0)
        {

                floor3();
        }
        }
}

        void delay (unsigned int ms)            // This loop is used for time taken by each led is
ON or OFF.
{
          unsigned int i, j;
          for (i=0; i<=ms; i++)                 // Here Loop indicates 50 millisecond.
```

```
            for (j=0; j<390; j++);
}

                    void floor1()
{
        if(ledb1==1)
        {
                    led1=1;
                      ledb1=0;
                         delay (50);
                         ledb2=1;
                    delay (50);
                    ledb2=0;
                    ledb3=1;
                    delay (50);
                    ledb3=0;
                    ledb4=1;
                    led1=0;
                                }
        else if (ledb6==1)
        {
                    led1=1;
                    ledb6=0;
                    ledb5=1;
                    delay(50);
                    ledb5=0;
                    ledb4=1;
                    led1=0;
        }

        else if (ledb8==1)
        {
                    led1=1;
                    ledb8=0;
                    ledb7=1;
                    delay(50);
                    ledb7=0;
                    ledb6=1;
                    delay(50);
                    ledb6=0;
                    ledb5=1;
                    delay(50);
                    ledb5=0;
                    ledb4=1;
                    led1=0;
        }
}
                            void floor2()
{
        if(ledb1==1)
        {
                    led2=1;
                      ledb1=0;
```

```
                delay (50);
                ledb2=1;
                delay (50);
            ledb2=0;
            ledb3=1;
             delay (50);
            ledb3=0;
            ledb4=1;
            delay (50);
            ledb4=0;
            ledb5=1;
            delay (50);
            ledb5=0;
            ledb6=1;
             led2=0;

}
else if (ledb4==1)
{
            led2=1;
            ledb4=0;
            ledb5=1;
            delay (50);
            ledb5=0;
            ledb6=1;
             led2=0;
}

else if (ledb8==1)
{
            led2=1;
            ledb8=0;
            ledb7=1;
            delay(50);
            ledb7=0;
            ledb6=1;
            led2=0;
                        }
            }
void floor3()
            {
                        if(ledb1==1)
                        {
                                    led3=1;
                                    ledb1=0;
                                    ledb2=1;
                                    delay(50);
                                    ledb2=0;
                                    ledb3=1;
                                    delay(50);
                                    ledb3=0;
                                    ledb4=1;
                                    delay(50);
```

```
                                ledb4=0;
                                ledb5=1;
                                delay(50);
                                ledb5=0;
                                ledb6=1;
                                delay(50);
                                ledb6=0;
                                ledb7=1;
                                delay(50);
                                ledb7=0;
                                ledb8=1;
                                led3=0;


                        }
                        else if (ledb4==1)
                        {
                                led3=1;
                                ledb4=0;
                                ledb5=1;
                                delay(50);
                                ledb5=0;
                                ledb6=1;
                                delay(50);
                                ledb6=0;
                                ledb7=1;
                                delay(50);
                                ledb7=0;
                                ledb8=1;
                                led3=0;
                        }
                        else if (ledb6==1)
                        {
                                led3=1;
                                ledb6=0;
                                ledb7=1;
                                delay(50);
                                ledb7=0;
                                ledb8=1;
                                led3=0;
                        }
                }

                        void floorground()
{
        if(ledb8==1)
        {
                led0=1;
                ledb8=0;
                ledb7=1;
                delay(50);
                ledb7=0;
                ledb6=1;
```

```
        delay(50);
        ledb6=0;
        ledb5=1;
        delay(50);
        ledb5=0;
        ledb4=1;
        delay(50);
        ledb4=0;
        ledb3=1;
        delay(50);
        ledb3=0;
        ledb2=1;
        delay(50);
        ledb2=0;
        ledb1=1;
        led0=0;
}
else if (ledb6==1)
{ led0=1;
                ledb6=0;
        ledb5=1;
        delay(50);
        ledb5=0;
        ledb4=1;
        delay(50);
        ledb4=0;
        ledb3=1;
        delay(50);
        ledb3=0;
        ledb2=1;
        delay(50);
        ledb2=0;
        ledb1=1;
        led0=0;
}
else if (ledb4==1)
{
        led0=1;
        ledb4=0;
        ledb3=1;
        delay(50);
        ledb3=0;
        ledb2=1;
        delay(50);
        ledb2=0;
        ledb1=1;
        led0=0;
}
}
```

Output: Elevator control circuit is designed for four floors and seen with help of LEDs.

Result: We have concluded that through the above method, how we can implement real life problems like Elevator controller using 8051 microcontroller.

Discussion:

1) Explain how Microcontroller is different from Microprocessor?
2) Explain difference in: if-else and case statement used in coding style?
3) Make a LED chaser using 8 LEDs?
4) Make a capacity based Elevator system?

EXP: B1

Aim: WAP in 8085 assembly language multiply two 8 Bit Numbers (04, 02) whose result is 16 bit.

Apparatus required: VINYTICS VMC-8501 kit

Procedure:

a) Get the 1st 8 bit numbers
b) Move the 1st 8it number to register 'B'
c) Get the 2nd 8 bit number
d) Move the 2nd 8 bit number to register 'C'
e) Initialize the accumulator as zero
f) Initialize the carry as zero
g) Add both register 'B' value as accumulator
h) Jump on if no carry
i) Increment carry by 1 if there is
j) Decrement the 2nd value and repeat from step 8, till the 2nd value becomes zero.
k) Store the multiplied value in accumulator
l) Move the carry value to accumulator
m) Store the carry value in accumulator

Program:

M. ADDRESS	HEX CODE	MNEMONICS		COMMENTS
		OPCODE	OPERAND	
2000H	3AH	LDA	3000H	Load direct A by 3000H content
2001H	00H			
2002H	30H			
2003H	47H	MOV	B,A	Move data from register A to B.
2004H	3AH	LDA	3001H	Load direct A by 3001H content
2005H	01H			
2006H	30H			
2007H	4FH	MOV	C,A	Move data from register A to C.
2008H	3EH	MVI	A, 00	Move data 00 into register A.
2009H	00H			
200AH	16H	MVI	D,00H	Move data 00 into register D.
200BH	00H			

200CH	80H	ADD	B	Add register B and accumulator
200DH	D2H	JNC	2011H	Jump if no carry to 2011H
200EH	11H			
200FH	20H			
2010H	14H	INR	D	Increment reg. pair D
2011H	0DH	DCR	C	Decrease in resister C
2012H	C2H	JNZ	200CH	Jump if not zero at address 200CH
2013H	0CH			
2014H	20H			
2015H	32H	STA	3002H	Store the result to 3002H
2016H	02H			
2017H	30H			
2018H	7AH	MOV	A,D	Move data from register D to A.
2019H	32H	STA	3003H	Store the result at the location M3003H.
201AH	03H			
201BH	30H			
201CH	76H	HLT		Stop the program

Output:

BEFORE EXECUTION		AFTER EXECUTION	
MEMORY ADDRESS	CONTENT	MEMORY ADDRESS	CONTENT
M3000H	04H	M3002H	08H
M3001H	02H	M3003H	00H

Result:

We have concluded that how arithmetic operation like multiplication can be implemented, and how we can get 16 bit result by checking carry flag.

Discussion:

1) List the major components of 8279 keyboard /display interface?
2) What is USART?
3) List the major components of 8251A programmable communication interface?
4) Give the various modes of 8254 timer?
5) Explain PUSH PSW, LXI SP, 16bit, JC, DCX SP instructions?

EXP: B2

Aim: WAP to convert packed BCD (29 H) to two ASCII numbers and store result in 4062H and 4063H locations.

Apparatus required: VINYTICS VMC-8501 kit

Program:

```
        MOV A, #29H
        MOV R2, A
        ANL A,#0FH
       ORL A,#30H
        MOV R6, A
       MOV A,R2
       ANL A, #0F0H
       RR A
       RR A
       RR A
       RR A
       ORL A, #30H
       MOV DPTR, #4062H
       MOV @DPTR, A
       INC DPTR
       MOV A,R6
       MOV @DPTR, A
HERE: SJMP HERE
```

Output:

BEFORE EXECUTION		AFTER EXECUTION	
ACCUMALATOR	CONTENT	MEMORY ADDRESS	CONTENT
ACCUMALATOR	29H	M4062H	39H
		M4063H	32H

Result:

We have checked and concluded that how packed BCD can be converted to ASCII standred that can be used in real time clock application.

Discussion:

1) What value must R4 have in order for the following instruction not to jump?
CJNE R4, #53, OVER

2) Find the contents of register an after execution of the following code.

```
CLR A
ORL A, #99H
CPL A
```

3) Explain the difference between a carry and an overflow.
4) What bit addresses are assigned to P1?