3.7 COMPARISON BETWEEN MICROPROCESSOR AND MICROCONTROLLER

S.No	Microprocessor	Microcontroller
1	A microprocessor is a general purpose device which is called a CPU	A microcontroller is a dedicated chip which is also called single chip computer.
2	A microprocessor do not contain onchip I/OPorts, Timers, Memories etc..	A microcontroller includes RAM, ROM, serial and parallel interface, timers, interrupt circuitry (in addition to CPU) in a single chip.
3	Microprocessors are most commonly used as the CPU in microcomputer systems	Microcontrollers are used in small, minimum component designs performing control-oriented applications.
4	Microprocessor instructions are mainly nibble or byte addressable	Microcontroller instructions are both bit addressable as well as byte addressable.
5	Microprocessor instruction sets are mainly intended for catering to large volumes of data.	Microcontrollers have instruction sets catering to the control of inputs and outputs.
6	Microprocessor based system design is complex and expensive	Microcontroller based system design is rather simple and cost effective
7	The Instruction set of microprocessor is complex with large number of instructions.	The instruction set of a Microcontroller is very simple with less number of instructions. For, ex: PIC microcontrollers have only 35 instructions**.**

8	A microprocessor has zero status flag	A microcontroller has no zero flag.

3.6 INSTRUCTION SET

The microcontroller 8051 instructions set includes 110 instructions, 49 of which are single byte instructions, 45 are two bytes instructions and 17 are three bytes instructions. The instructions format consists of a function mnemonic followed by destination and source field.

ADDRESSING MODES OF 8051 :

The way in which the data operands are accessed by different instructions is known as the addressing modes. There are various methods of denoting the data operands in the instruction. The 8051 microcontroller supports mainly 5 addressing modes. They are

1. Immediate addressing mode
2. Direct Addressing mode
3. Register addressing mode
4. Register Indirect addressing mode
5. Indexed addressing mode

Immediate addressing mode :

The addressing mode in which the data operand is a constant and it is a part of the instruction itself is known as Immediate addressing mode. Normally the data must be preceded by a # sign. This addressing mode can be used to transfer the data into any of the registers including DPTR.

Ex: MOV A , # 27 H : The data (constant) 27 is moved to the accumulator register

ADD R1 ,#45 H : Add the constant 45 to the contents of the accumulator

MOV DPTR ,# 8245H :Move the data 8245 into the data pointer register.

MOV P1,#21 H

Direct addressing mode:

The addressing mode in which the data operand is in the RAM location (00 -7FH) and the address of the data operand is given in the instruction is known as Direct addressing mode. The direct addressing mode uses the lower 128 bytes of Internal RAM and the SFRs

MOV R1, 42H : Move the contents of RAM location 42 into R1 register

MOV 49H,A : Move the contents of the accumulator into the RAM location 49.

ADD A, 56H : Add the contents of the RAM location 56 to the accumulator

Register addressing mode :

The addressing mode in which the data operand to be manipulated lies in one of the registers is known as register addressing mode.

MOV A,R0 : Move the contents of the register R0 to the accumulator

ADD A,R6 :Add the contents of R6 register to the accumulator

MOV P1, R2 : Move the contents of the R2 register into port 1

MOVR5, R2: This is invalid .The data transfer between the registers is not allowed.

Register Indirect addressing mode :

The addressing mode in which a register is used as a pointer to the data memory block is known as Register indirect addressing mode.

MOV A,@ R0 :Move the contents of RAM location whose address is in R0 into A (accumulator)

MOV @ R1 , B : Move the contents of B into RAM location whose address is held by R1

When R0 and R1 are used as pointers, they must be preceded by @ sign,One of the advantages of register indirect addressing mode is that it makes accessing the data more dynamic than static as in the case of direct addressing mode.

Indexed addressing mode :

This addressing mode is used in accessing the data elements of lookup table entries located in program ROM space of 8051.

Ex : MOVC A,@ A+DPTR

The 16-bit register DPTR and register A are used to form the address of the data element stored in on-chip ROM. Here C denotes code .In this instruction the contents of A are added to the 16-bit DPTR register to form the 16-bit address of the data operand.

INSTRUCTION SET OF 8051 MICROCONTROLLER

All members of the 8051 family execute the same instructions set. The 8051 instructions set is optimized for 8-bit content application. The Intel 8051 has excellent and most powerful instructions set offers possibilities in control area, serial Input/Output, arithmetic, byte and bit manipulation.

It has 111 instructions they are

- 49 single byte instructions

- 45 two bytes instructions
- 17 three bytes instructions

The- instructions set is divided into four groups, they are

- Data transfer instructions
- Arithmetic instructions
- Logical instructions
- Call and Jump instructions

DATA TRANSFER INSTRUCTIONS

This instruction copies the contents of the source location to the destination location. The contents of the source location are unchanged.

Instruction format	Algorithm	Example	Functon
MOVA, Rn	A = Rn	MOV A, R1	Move byte form register Rn to accumulator
MOV A, direct	A = direct	MOV A, 40H	Move byte from direct address to accumulator
MOV A, @ Ri	A=[[Ri]]	MOVA,@ R0	Move the content of memory location to accumulator
MOV A, # data	A = # data	MOV A, #31 H	Move immediate data to accumulator
MOV Rn, A	Rn = A	MOV R5, A	Move data from accumulator to register Rn
MOV Rn, direct	Rn = direct	MOV R3, 30H	Move data from direct address to register Rn.
MOV Rn, # data	Rn = # data	MOV R7, #20H	Move immediate data to register Rn.
MOV direct, A	direct =A	MOV 80H, A	Move data from accumulator to direct address
MOV direct, Rn	direct = Rn	MOV 30H, R5	Move data from register to direct address

MOV direct, direct	direct = direct	MOV 20H, 30H	Move data form source direct address to the destination direct address.
MOV direct, @Ri	direct = [[Ri]]	MOV 20H, @R1	Move data from address specified in register Ri to direct address.
MOV direct, #data	direct = #data	MOV 10H, #10H	Move immediate data to direct address
MOV @Ri, A	[[Ri]] = A	MOV @ R0,A	Move data form accumulator to memory location pointed by Ri

ARITHMETIC INSTRUCTIONS:

Mnemonic	Example	Description
ADD A, Rn	ADD A, R0	This instruction will add the byte in register Rn of the selected register bank with the byte in accumulator. The result is contained in the accumulator
ADD A, direct	ADD A, 20H	This instruction will add the contents of the memory location whose direct address is specified in the instruction with the accumulator contents. The result of addition will he stored in the accumulator.
ADD A, @ Ri	ADD A, @ R0	This instruction will add the contents of memory location whose address is pointed by register Ri of the selected register bank with contents of the accumulator. The result of addition is stored in the accumulator
ADD A, # data	ADD A, # 30H	This instruction will add the immediate 8 bit data with data in the accumulator. The result of addition is stored in the accumulator.

Mnemonics	Addressing mode	Example	Description
ADDC A, Rn	Register addressing	ADDC A, Rl	This instruction will add the contents of accumulator with the contents of register Rn of the selected register bank and carry flag. The result of addition is stored in accumulator.
ADDC A, direct	Direct addressing	ADDC A, 10H	This instruction will add the contents of memory location whose direct address is specified in the instruction with the contents of accumulator and carry. The result of addition is stored in the accumulator.
ADDC A, @Ri	Register Indirect	ADDC A, @R0	This instruction will add the contents of memory location pointed by register Ri of selected register bank with the accumulator and earn' flag. The result is stored in accumulator.
ADDC A, #data	Immediate addressing	ADDC A, #40H	This instruction will add the contents of accumulator with immediate data specified in the instruction along with carry.
ADDC A, Rn	Register addressing	ADDC A, Rl	This instruction will add the contents of accumulator with the contents of register Rn of the selected register bank and carry flag. The result of addition is stored in accumulator.

ADDC A, direct	Direct addressing	ADDC A, 10H	This instruction will add the contents of memory location whose direct address is specified in the instruction with the contents of accumulator and carry. The result of addition is stored in the accumulator.
ADDC A, @Ri	Register Indirect	ADDC A, @R0	This instruction will add the contents of memory location pointed by register Ri of selected register bank with the accumulator and earn' flag. The result is stored in accumulator.
ADDC A, #data	Immediate addressing	ADDC A, #40H	This instruction will add the contents of accumulator with immediate data specified in the instruction along with carry.

LOGICAL INSTRUCTIONS:

Mnemonics	Addressing mode	Example	Description
ANL A, Rn	Register addressing	ANL A, R5	This instruction will perform bit wise logical AND operation between the contents of accumulator and register Rn of the selected register bank. The result will be stored in the accumulator
ANL A, direct	Direct addressing	ANL A, 70H	This instruction will bit wise logically AND the contents of accumulator with the contents of memory location whose direct address is specified in the instruction. The result will be stored in the accumulator
ANL direct, A	Direct addressing	AXL 30H, A	This instruction will bit wise logically AND the contents of memory location whose direct address is specified in the instruction with the contents of accumulator. The result will be stored

			in the memory location whose direct address is specified in the instruction.
ANL A, ®Ri	Register Indirect Addressing	ANL A, @R1	This instruction will bit wise logically AND the contents of accumulator with the contents of memory location pointed by register Ri of the selected register bank. The result will be stored
ANL A, #data	Immediate addressing	ANL A, #57H	This instruction will bit wise logically AND the contents of accumulator with the immediate data specified in the instruction. The result will be stored in the accumulator.
ANL direct, #data	Immediate addressing	ANL 54H, #33H	This instruction will bit wise logically AND the contents of memory location whose direct address is specified in the instruction with the contents of with the immediate data specified in the instruction. The result will be stored in the memory location whose direct address is specified in the instruction.

CLRA

- This instruction will clear all the bits of accumulator to zero.

CPLA

- This instruction will complements all the bits (l's complement) of the accumulator.

RLA

- This instruction will rotate the eight bits in the accumulator by one bit to the left.
- Addressing mode: Register Specific addressing mode.

RLCA

- This instruction will rotate the eight bits in the accumulator and the carry flag together by one bit to the left
- Addressing mode: Register Specific addressing mode.

RRA

- This instruction will rotate the eight bits in the accumulator by one bit to the right.
- Addressing mode: Register Specific addressing mode.

RRCA

- This instruction will rotate the eight bits in the accumulator and the carry flag together by one bit to the right.
- Addressing mode: Register Specific addressing mode.

SWAP A

This instruction interchanges the low order and high order nibbles of the accumulator. Operation:

- Addressing mode: Register Specific addressing mode.

3.1 FUNCTIONAL BUILDING BLOCKS & HARDWARE ARCHITECTURE

Introduction :

A decade back the process and control operations were totally implemented by the Microprocessors only. But now a days the situation is totally changed and it is occupied by the new devices called Microcontroller. The development is so drastic that we can't find any electronic gadget without the use of a microcontroller. This microcontroller changed the embedded system design so simple and advanced that the embedded market has become one of the most sought after for not only entrepreneurs but for design engineers also.

Microcontroller

A single chip computer or A CPU with all the peripherals like RAM, ROM, I/O Ports, Timers , ADCs etc... on the same chip. For ex: Motorola's 6811, Intel's 8051, Zilog's Z8 and PIC 16X etc…

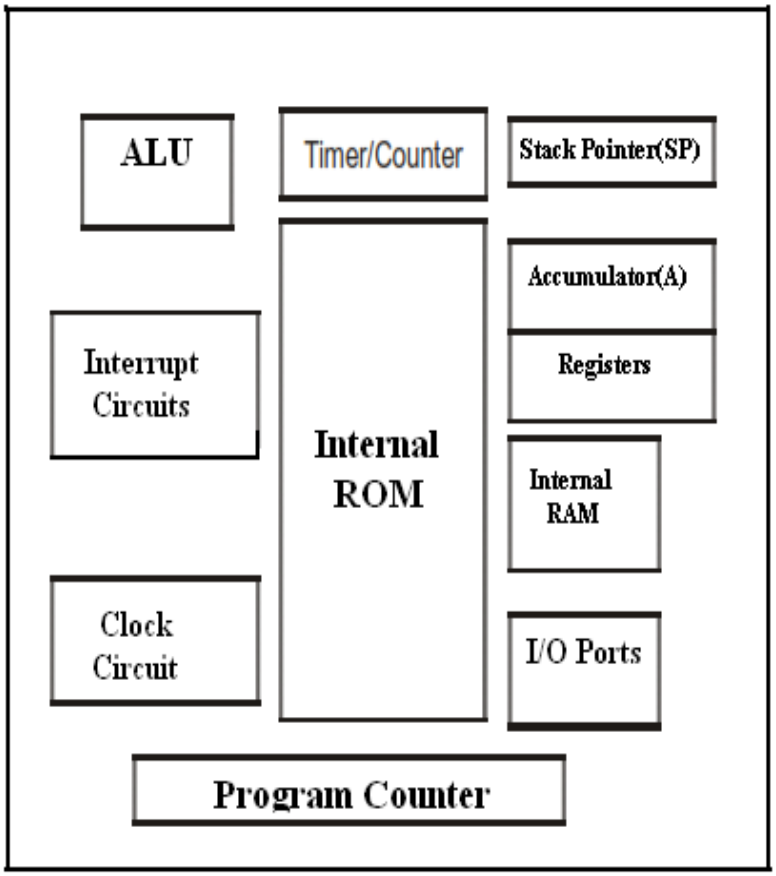

Figure 3.1.1 Block Diagram of a Microcontroller

[Source: "Microprocessor Architecture Programming and Application" by R.S. Gaonkar, page-]

S.No	Microprocessor	Microcontroller
1	A microprocessor is a general purpose device which is called a CPU	A microcontroller is a dedicated chip which is also called single chip computer.
2	A microprocessor do not contain onchip I/Ports, Timers, Memories etc..	A microcontroller includes RAM, ROM, serial and parallel interface, timers, interrupt circuitry (in addition to CPU) in a single chip.
3	Microprocessors are most commonly used as the CPU in microcomputer systems	Microcontrollers are used in small, minimum component designs performing control-oriented applications.
4	Microprocessor instructions are mainly nibble or byte addressable	Microcontroller instructions are both bit addressable as well as byte addressable.
5	Microprocessor instruction sets are mainly intended for catering to large volumes of data.	Microcontrollers have instruction sets catering to the control of inputs and outputs.
6	Microprocessor based system design is complex and expensive	Microcontroller based system design is rather simple and cost effective
7	The Instruction set of microprocessor is complex with large number of instructions.	The instruction set of a Microcontroller is very simple with less number of instructions. For, ex:

		PIC microcontrollers have only 35 instructions.
8	A microprocessor has zero status flag	A microcontroller has no zero flag.

Evolution of Microcontrollers :

The first microcontroller TMS1000 was introduced by Texas Instrumentsin the year 1974. In the year 1976, Motorola designed a Microprocessor chip called 6801 which replaced its earlier chip 6800 with certain add-on chips to make a computer. This paved the way for the new revolution in the history of chip design and gave birth to a new entity called "Microcontroller". Later the Intel company produced its first Microcontroller 8048 with a CPU and 1K bytes of EPROM, 64 Bytes of RAM an 8-Bit Timer and 27 I/O pins in 1976. Then followed the most popular controller 8051 in the year 1980 with 4K bytes of ROM,128 Bytes of RAM , a serial port, two 16-bit Timers , and 32 I/O pins. The 8051 family has many additions and improvements over the years and remains a most acclaimed tool for today's circuit designers. INTEL introduced a 16 bit microcontroller 8096 in the year 1982 . Later INTEL introduced 80c196 series of 16-bit Microcontrollers for mainly industrial applications. Microchip, another company has introduced an 8-bit Microcontroller PIC 16C64 in the year 1985.The 32-bit microcontrollers have been developed by IBM and Motorola. MPC 505 is a 32-bit RISC controller of Motorola. The 403 GA is a 32 -bit RISC embedded controller of IBM.

In recent times ARM company (Advanced RISC machines) has developed and introduced 32 bit controllers for high-end application devices like mobiles , Ipods .

Microcontroller Development Tools:

To develop an assembly language program we need certain program development tools. An assembly language program consists of Mnemonics which are nothing but short abbreviated English instructions given to the controller.The various development tools required for Microcontroller programming are explained below.

1. Editor :

An Editor is a program which allows us to create a file containing the assembly language statements for the program. Examples of some editors are PC write Wordstar.

As we type the program the editor stores the ACSII codes for the letters and numbers in successive RAM locations. If any typing mistake is done editor will alert us to correct it. If we leave out a program statement an editor will let you move everything down and insert a line. After typing all the program we have to save the program . This we call it as source file. The next step is to process the source file with an assembler.

Ex: Sample. asm

2.Assembler :

An Assembler is used to translate the assembly language mnemonics into machine language(i.e binary codes). When you run the assembler it reads the source file of your program from where you have saved it. The assembler generates a filee with the extension .hex. This file consists of hexadecimal values encoding a sequence of data and their starting offset or absolute address.

3.Compiler : A compiler is a program which converts the high level language program like "C" into binary or machine code. Using high level languages it is easy to manage complex data structures which are often required for data manipulation. Because of its ease , flexibility and debug options now a days the compilers have become very popular in the market. Compilers like Keil ,Ride and IAR workbench are very popular.

3. Debugger/Simulator :

A debugger is a program which allows execute the program, and troubleshoot or debug it. The debugger allows to look into the contents of registers and memory locations after the program runs. We can also change the contents of registers and memory locations and rerun the program. Some debuggers allows to stop the program after each instruction so that you can check or alter memory and register contents. This is called single step debug. A debugger also allows to set a breakpoint at any point in the program. If we insert a break point , the debugger will run the program up to the instruction where the breakpoint is put and then stop the execution.

A simulator is a software program which virtually executes the instructions similar to a microcontroller and shows the results. This will help in evaluating the results without committing any errors. By doing so we can detect the possible logic errors.

INTEL 8051 Micrcontroller :

The 8051 microcontroller is a very popular 8-bit microcontroller introduced by Intel in the year 1981 and it has become almost the academic standard now a days. The 8051 is based on an 8-bit CISC core with Harvard architecture. Its 8-bit architecture is optimized for control applications with extensive Boolean processing. It is available as a 40-pin DIP chip and works at +5 Volts DC. The salient features of 8051 controller are given below.

Salient Features :

The salient features of 8051 Microcontroller are

i. 4 KB on chip program memory (ROM or EPROM)).

ii. 128 bytes on chip data memory(RAM).

iii. 8-bit data bus

iv. 16-bit address bus

v. 32 general purpose registers each of 8 bits

vi. Two -16 bit timers T0 and T1

vii. Five Interrupts (3 internal and 2 external).

ix. Four Parallel ports each of 8-bits (PORT0, PORT1,PORT2,PORT3) with a total of 32 I/O lines.

x. One 16-bit program counter and One 16-bit DPTR (data pointer)

xi. One 8-bit stack pointer

xii. One Microsecond instruction cycle with 12 MHz Crystal.

xiii. One full duplex serial communication port.

Architecture & Block Diagram of 8051 Microcontroller:

The architecture of the 8051 microcontroller can be understood from the block diagram. It has Harward architecture with RISC (Reduced Instruction Set Computer) concept. The block diagram of 8051 microcontroller is shown in Fig 3. below1.It consists of an 8-bit ALU, one 8-bit PSW(Program Status Register), A and B registers , one 16-bit Program counter , one 16-bit Data pointer register(DPTR),128 bytes of RAM and 4kB of ROM and four parallel I/O ports each of 8-bit width. 8051 has 8-bit ALU which can perform all the 8-bit arithmetic and logical operations in one machine cycle. The ALU is associated with two registers A & B

A and B Registers:

The A and B registers are special function registers which hold the results of many arithmetic and logical operations of 8051.The A register is also called the Accumulator and as it's name suggests, is used as a general register to accumulate the results of a large number of instructions. By default it is used for all mathematical operations and also data transfer operations between CPU and any external memory.

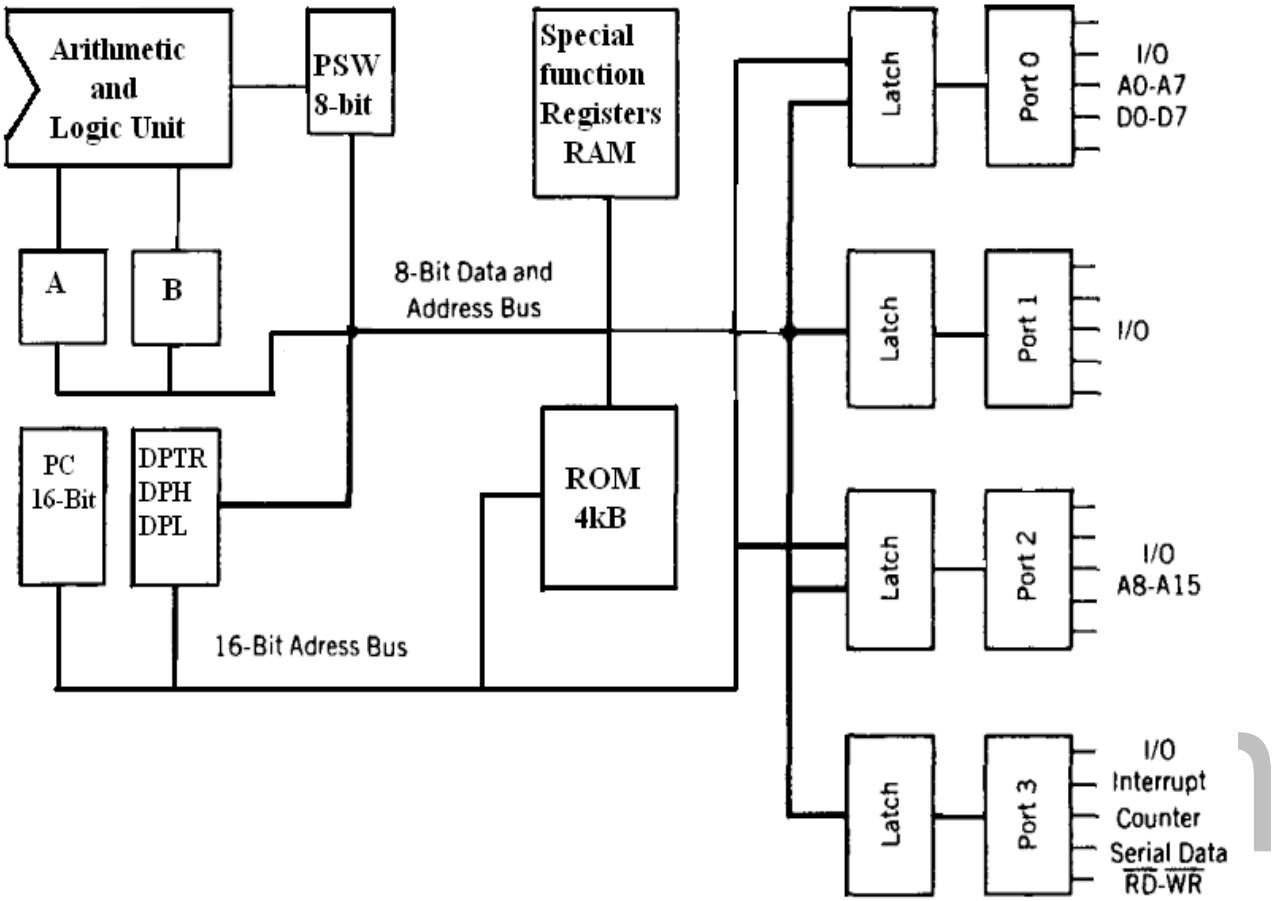

Figure 3.1.2 Block Diagram of 8051 Microcontroller

[Source: "Microprocessor Architecture Programming and Application" by R.S. Gaonkar, page-]

The R registers:

The "R" registers are a set of eight registers that are named R0, R1, etc. up to and including R7. These registers are used as auxiliary registers in many operations. The "R" registers are also used to temporarily store values.

Program Counter(PC):

8051 has a 16-bit program counter. The program counter always points to the address of the next instruction to be executed. After execution of one instruction the program counter is incremented to point to the address of the next instruction to be

executed. It is the contents of the PC that are placed on the address bus to find and fetch the desired instruction. Since the PC is 16-bit width ,8051 can access program addresses from 0000H to FFFFH, a total of 6kB of code.

Stack Pointer Register (SP) : It is an 8-bit register which stores the address of the stack top. i.e the Stack Pointer is used to indicate where the next value to be removed from the stack should be taken from. When a value is pushed onto the stack, the 8051 first increments the value of SP and then stores the value at the resulting memory location. Similarly, when a value is popped off the stack, the 8051 returns the value from the memory location indicated by SP, and then decrements the value of SP. Since the SP is only 8-bit wide it is incremented or decremented by two. SP is modified directly by the 8051 by six instructions: PUSH, POP, ACALL, LCALL, RET, and RETI. It is also used intrinsically whenever an interrupt is triggered.

STACK in 8051 Microcontroller:

The stack is a part of RAM used by the CPU to store information temporarily. This information may be either data or an address. The CPU needs this storage area as there are only limited number of registers. The register used to access the stack is called the Stack pointer which is an 8-bit register. So, it can take values of 00 to FF When the8051 is powered up, the SP register contains the value 07.i.e the RAM location value 08 is the first location being used for the stack by the 8051 controller

There are two important instructions to handle this stack.One is the PUSH and the Other is the POP. The loading of data from CPU registers to the stack is done by PUSH and the loading of the contents of the stack back into aCPU register is done by POP.

```
EX: MOV R6 ,#35 H
    MOV R1 ,#21 H
    PUSH 6
    PUSH 1
```

In the above instructions the contents of the Registers R6 and R1 are moved to stack and they occupy the 08 and 09 locations of the stack. Now the contents of the SP are incremented by two and it is 0A

Similarly, POP 3 instruction pops the contents of stack into R3 register. Now the contents of the SP is decremented by 1

In 8051 the RAM locations 08 to 1F (24 bytes) can be used for the Stack.In any program if we need more than 24 bytes of stack ,we can change the SP point to RAM locations 30-7F H.this can be done with the instruction MOV SP,# XX.

Data Pointer Register(DPTR):

It is a 16-bit register which is the only user-accessible. DPTR, as the name suggests, is used to point to data. It is used by a number of commands which allow the 8051 to access external memory. When the 8051 accesses external memory it will access external memory at the address indicated by DPTR. This DPTR can also be used as two 8-registers DPH and DPL.

Program Status Register (PSW):

The 8051 has a 8-bit PSW register which is also known as Flag register. In the 8-bit register only 6-bits are used by 8051.The two unused bits are user definable bits. In the 6-bits four of them are conditional flags. They are Carry –CY, Auxiliary Carry-AC, Parity-P, and Overflow-OV. These flag bits indicate some conditions that resulted after an instruction was executed.

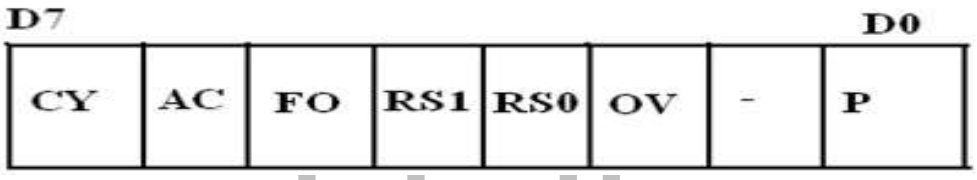

Figure 3.1.2 Program Status Word of 8051 Microcontroller

[Source: "Microprocessor Architecture Programming and Application" by R.S. Gaonkar, page-]

The bits PSW3 and PSW4 are denoted as RS0 and RS1 and these bits are used th select the bank registers of the RAM location. The meaning of various bits of PSW register is shown below.

CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry Flag
FO	PSW.5	Flag 0 available for general purpose .
RS1	PSW.4	Register Bank select bit 1
RS0	PSW.3	Register bank select bit 0
OV	PSW.2	Overflow flag
---	PSW.1	User difinable flag
P	PSW.0	Parity flag .set/cleared by hardware.

The selection of the register Banks and their addresses are given below.

RS1	RS0	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

3.4 I/O PORTS AND DATA TRANSFER CONCEPTS

Parallel I /O Ports :

The 8051 microcontroller has four parallel I/O ports , each of 8-bits .So, it provides the user 32 I/O lines for connecting the microcontroller to the peripherals. The four ports are P0 (Port 0), P1(Port1) ,P2(Port 2) and P3 (Port3). Upon reset all the ports are output ports. In order to make them input, all the ports must be set i.e a high bit must be sent to all the port pins. This is normally done by the instruction "SETB".

```
Ex:     MOV A,#0FFH     ; A =  FF
        MOV P0,A        ; make P0 an input port
```

PORT 0:

Port 0 is an 8-bit I/O port with dual purpose. If external memory is used, these port pins are used for the lower address byte address/data (AD0-AD7), otherwise all bits of the port are either input or output.. Unlike other ports, Port 0 is not provided with pull-up resistors internally ,so for PORT0 pull-up resistors of nearly 10k are to be connected externally as shown.

Dual role of port 0:

Port 0 can also be used as address/data bus(AD0-AD7), allowing it to be used for both address and data. When connecting the 8051 to an external memory, port 0 provides both address and data. The 8051 multiplexes address and data through port 0 to save the pins. ALE indicates whether P0 has address or data. When ALE = 0, it provides data D0-D7, and when ALE =1 it provides address and data with the help of a 74LS373 latch.

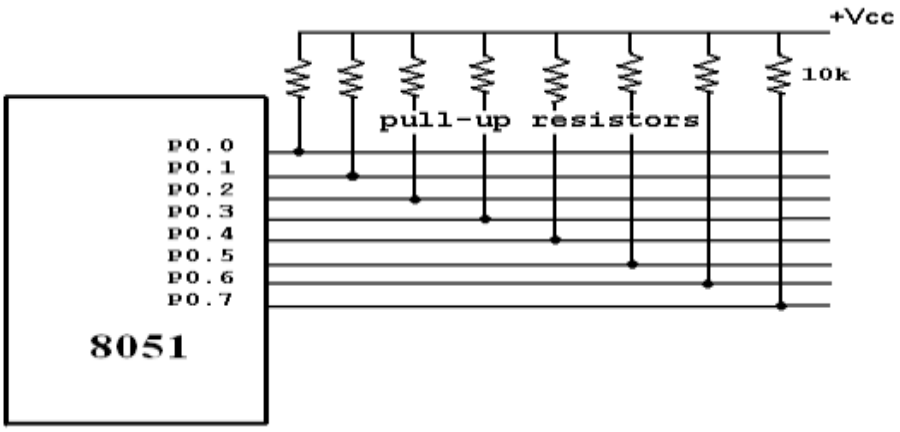

Figure 3.4.1 Dual role of port 0

[Source: "Microprocessor Architecture Programming and Application" by R.S. Gaonkar, page-]

Port 1:

Port 1 occupies a total of 8 pins (pins 1 through 8). It has no dual application and acts only as input or output port. In contrast to port 0, this port does not need any pull-up resistors since pull-up resistors connected internally. Upon reset, Port 1 is configured as an output port. To configure it as an input port, port bits must be set i.e a high bit must be sent to all the port pins. This is normally done by the instruction "SETB".

```
Ex:   MOV A, #0FFH; A=FF HEX
      MOV P1, A; make P1 an input port by writing 1's to all of its pins
```

Port 2:

Port 2 is also an eight-bit parallel port. (pins 21- 28). It can be used as input or output port. As this port is provided with internal pull-up resistors it does not need any external pull-up resistors. Upon reset, Port 2 is configured as an output port. If the port is to be used as input port, all the port bits must be made high by sending FF to the port. For

```
Ex:   MOV A, #0FFH      ; A=FF hex
      MOV P2, A         ; make P2 an input port by writing all 1's to it
```

Dual role of port 2:

Port2 lines are also associated with the higher order address lines A8-A15. In systems based on the 8751, 8951, and DS5000, Port2 is used as simple I/O port. But, in 8031-based systems, port 2 is used along with P0 to provide the 16-bit address for the external memory. Since an 8031 is capable of accessing 64K bytes of external memory, it needs a path for the 16 bits of the address. While P0 provides the lower 8 bits via A0-A7, it is the job of P2 to provide bits A8-A15 of the address. In other words, when 8031 is connected to external memory, Port 2 is used for the upper 8 bits of the 16-bit address, and it cannot be used for I/O operations.

PORT 3:

Port3 is also an 8-bit parallel port with dual function. (pins 10 to 17). The port pins can be used for I/O operations as well as for control operations. The details of these additional operations are given below in the table. Port 3 also do not need any external pull-up resistors as they are provided internally similar to the case of Port2 & Port 1. Upon reset port 3 is configured as an output port. If the port is to be used as input port, all the port bits must be made high by sending FF to the port.

Port 1: Ex: MOV A, #0FFH ; A= FF hex
MOV P3, A ; make P3 high (8) It has by default all 1's on it

Port 1 occupies a total of 8 pins (pins 1 through 8). It has no dual application and can only function as input port.

Alternate Functions of Port 3: In contrast to port 0, this port does not need any pull-up resistors since it has pull-up resistors connected internally. Port 3 is configured as an input port. To configure it as an input port, port bits must be set to a high bit must be used for Timer/Counter 0 and 1 and P3.6 and P3.7 are used to provide the write and read signals of external memories connected in 8031 based systems

S.No	Port 3 bit	Pin No	Function
1	P3.0	10	RxD
2	P3.1	11	TxD
3	P3.2	12	$\overline{INT0}$
4	P3.3	13	$\overline{INT1}$
5	P3.4	14	T0
6	P3.5	15	T1
7	P3.6	16	$\overline{WR}$
8	P3.7	17	$\overline{RD}$

Serial communication

Serial communication uses only one or two data lines to transfer data and is generally used for long distance communication. In serial communication the data is sent as one bit at a time in a timed sequence on a single wire. Serial Communication takes place in two methods, Asynchronous data Transfer and Synchronous Data Transfer.

Asynchronous data transfer:

It allows data to be transmitted without the sender having to send a clock signal to the receiver. Instead, special bits will be added to each word in order to synchronize the sending and receiving of the data. When a word is given to the UART for Asynchronous transmissions, a bit called the "Start Bit" is added to the beginning of each word that is to be transmitted. The Start Bit is used to alert the receiver that a word of data is about to be sent, and to force the clock in the receiver into synchronization with the clock in the transmitter.

After the Start Bit, the individual bits of the word of data are sent. Here each bit in the word is transmitted for exactly the same amount of time as all of the other bits. When the entire data word has been sent, the transmitter may add a Parity Bit that the transmitter generates. The Parity bit may be used by the receiver to perform simple error checking. Then at least one Stop Bit is sent by the transmitter. If the Stop Bit does not appear when it is supposed to, the UART considers the entire word to be corrupted and will report a Framing Error.

Port 1:

Port 1 occupies a total of 8 pins (pins 1 through 8). It has no dual application and acts only as input or output port. In contrast to port 0, this port does not need any pull-up resistors since pull-up resistors connected internally. Upon reset, Port 1 is configured as an output port. To configure it as an input port, port bits must be set i.e a high bit must be sent to all the port pins. This is normally done by the instruction "SETB".

Baud rate is a measurement of transmission speed in asynchronous communication, it represents the number of bits/sec that are actually being sent over the serial link. The Baud count includes the overhead bits Start, Stop and Parity that are generated by the sending UART and removed by the receiving UART.

Synchronous data transfer:

In this method the receiver knows when to "read" the next bit coming from the sender. This is achieved by sharing a clock between sender and receiver. In most forms of serial Synchronous communication, if there is no data available at a given time to transmit, a fill character will be sent instead so that data is always being transmitted. Synchronous communication is usually more efficient because only data bits are transmitted between sender and receiver, however it will be costlier because extra wiring and control circuits are required to share a clock signal between the sender and receiver. Devices that use serial cables for their communication are split into two categories.

1. DTE (Data Terminal Equipment). Examples of DTE are computers, printers & terminals.
2. DCE (Data Communication Equipment). Example of DCE is modems.

Parallel Data Transfer:

Parallel communication uses multiple wires (bus) running parallel to each other, and can transmit data on all the wires simultaneously. i.e all the bits of the byte are transmitted at a time. So, speed of the parallel data transfer is extremely high compared to serial data transfer. An 8-bit parallel data transfer is 8-times faster than serial data transfer. Hence with in the computer all data transfer is mainly based on Parallel data transfer. But only limitation is due to the high cost, this method is limited to only short distance communications.

Port 1:

Port 1 occupies a total of 8 pins (pins 1 through 8). It has no dual application and acts only as input or output port. In contrast to port 0, this port does not need any pull-up resistors since pull-up resistors connected internally. Upon reset, Port 1 is configured as an output port. To configure it as an input port, port bits must be set i.e a high bit must be sent to all the port pins. This is normally done by the Instruction "SETB".

S.No	Serial Communication	Parallel Communication
1	Data istransmitted bit after the bit in a single line	Data is transmitted simultaneously through group of lines(Bus)
2	Data congestion takes place	No Data congestion
3	Low Speed transmission	High speed transmission
4	Implementation of serial links is not an easy task.	Parallel data links are easily implemented in hardware
5.	In terms of transmission channel costs such as data bus cable length, data bus buffers, interface connectors, it is less expensive	It is more expensive
6	No , crosstalk problem	Crosstalk creates interference between the parallel lines.
7	No effect of inter symbol interference and noise	Parallel ports suffer extremely from inter-symbol interference (ISI) and noise, and therefore the data can be corrupted over long distances.
8	The bandwidth of serial wires is much higher.	The bandwidth of parallel wires is much lower.
9	Serial interface is more flexible to upgrade , without changing the hardware	Parallel data transfer mechanism rely on hardware resources and hence not flexible to upgrade.
10	Serial communication work effectively even at high frequencies.	Parallel buses are hard to run at high frequencies.

3.5 Interrupts

Port 1 consists of 8 pins (pins 1 through 8). It has no alternate functions. Port 1 can only be used as input or output. In contrast to port 0, no pull-up resistors are required for port 1. Upon reset, port 1 is configured as an output port. To configure it as an input port, port bits must be set high by the programmer.

Port 1 pins do not have any alternate functions.

An interrupt is a signal to the microcontroller that a device needs its service. The program which is associated with the interrupt is called the interrupt service routine (ISR). Upon receiving an interrupt signal, the microcontroller finish current instruction and save the PC on stack. Jumps to a fixed location in memory to begin execution of ISR. Starts to execute the interrupt service routine until RETI (return from interrupt)Upon executing the RETI the microcontroller returns to the place where it was interrupted. Get pop PC from stack.

The 8051 microcontroller has FIVE interrupts in addition to Reset. They are

- Timer 0 overflow Interrupt
- Timer 1 overflow Interrupt
- External Interrupt 0(INT0)
- External Interrupt 1(INT1)
- Serial Port events (buffer full, buffer empty, etc) Interrupt

Each interrupt has a specific place in code memory where program execution (interrupt service routine) begins.

- External Interrupt 0 : 0003 H
- Timer 0 overflow : 000B H
- External Interrupt 1 : 0013 H
- Timer 1 overflow : 001B H
- Serial Interrupt : 0023 H

Upon reset all Interrupts are disabled & do not respond to the Microcontroller. These interrupts must be enabled by software in order for the Microcontroller to respond to them. This is done by an 8-bit register called Interrupt Enable Register (IE).

Interrupt Enable Register :

EA	—	ET2	ES	ET1	EX1	ET0	EX0

Port 1

- Port 1 occupies a total of 8 pins (pins 1 through 8). It has no dual application and acts only as input or output port. In contrast to port 0, this port does not need any pull-up resistors since pull-up resistors connected internally. Upon reset, Port 1 is configured as an output port. To configure it as an input port, port bits must be set i.e a high bit must be sent to all the port pins. This is normally done by the instruction “SETB”.

- EA : Global enable/disable. To enable the interrupts this bit must be set High.
- — : Undefined reserved for future use.
- ET2 : Enable /disable Timer 2 overflow interrupt.
- ES : Enable/disable Serial port interrupt.
- ET1 : Enable /disable Timer 1 overflow interrupt.
- EX1 : Enable/disable External interrupt1.
- ET0 : Enable /disable Timer 0 overflow interrupt.
- EX0 : Enable/disable External interrupt0

Upon reset the interrupts have the following priority.(Top to down). The interrupt with the highest PRIORITY gets serviced first.

1. External interrupt 0 (INT0)
2. Timer interrupt0 (TF0)
3. External interrupt 1 (INT1)
4. Timer interrupt1 (TF1)
5. Serial communication (RI+TI)

Priority can also be set to “high” or “low” by 8-bit IP register.- Interrupt priority register

—	—	PT2	PS	PT1	PX1	PT0	PX0

IP.7: reserved

IP.6: reserved

IP.5: Timer 2 interrupt priority bit (8052 only)

IP.4: Serial port interrupt priority bit

IP.3: Timer 1 interrupt priority bit

IP.2: External interrupt 1 priority bit

IP.1: Timser 0 interrupt priority bit

IP.0: External interrupt 0 priority bit

3.3 MEMORY ORGANIZATION

Memory organization :

The 8051 microcontroller has 128 bytes of Internal RAM and 4kB of on chip ROM. The RAM is also known as Data memory and the ROM is known as program memory. The program memory is also known as Code memory. This Code memory holds the actual 8051 program that is to be executed. In 8051 this memory is limited to 64K .Code memory may be found on-chip, as ROM or EPROM. It may also be stored completely off-chip in an external ROM or, more commonly, an external EPROM. The 8051 has only 128 bytes of Internal RAM but it supports 64kB of external RAM. As the name suggests, external RAM is any random access memory which is off-chip. Since the memory is off-chip it is not as flexible interms of accessing, and is also slower. For example, to increment an Internal RAM location by 1,it requires only 1 instruction and 1 instruction cycle but to increment a 1-byte value stored in External RAM requires 4 instructions and 7 instruction cycles. So, here the external memory is 7 times slower.

Port 1:

Port 1 occupies a total of 8 pins (pins 1 through 8). It has no dual application and acts only as input or output port. In contrast to port 0, this port does not need any pull-up resistors since pull-up resistors connected internally. Upon reset, Port 1 is configured as an output port. To configure it as an input port, 8 bits of this port must be set i.e. high bits must be sent to all the port pins. This is normally done by the instruction "SETB".

Internal RAM OF 8051 :

This Internal RAM is found on-chip on the 8051 .So it is the fastest RAM available, and it is also the most flexible in terms of reading, writing, and modifying it's contents. Internal RAM is volatile, so when the 8051 is reset this memory is cleared. The 128 bytes of internal RAM is organized as below.

(i) Four register banks (Bank0,Bank1, Bank2 and Bank3) each of 8-bits (total 32 bytes). The default bank register is Bank0. The remaining Banks are selected with the help of RS0 and RS1 bits of PSW Register.

(ii) 16 bytes of bit addressable area and

(iii) 80 bytes of general purpose area (Scratch pad memory) as shown in the diagram below. This area is also utilized by the microcontroller as a storage area for the operating stack.

The 32 bytes of RAM from address 00 H to 1FH are used as working registers organized as four banks of eight registers each.The registers are named as R0-R7 .Each register can be addressed by its name or by its RAM address.

EX : MOV A, R7 or MOV R7,#05H

Port 1:

Port 1 occupi lual application and acts only as input oı ot need any pull-up resistors since pull- t 1 is configured as an output port. To c i.e a high bit must be sent to all the po SETB".

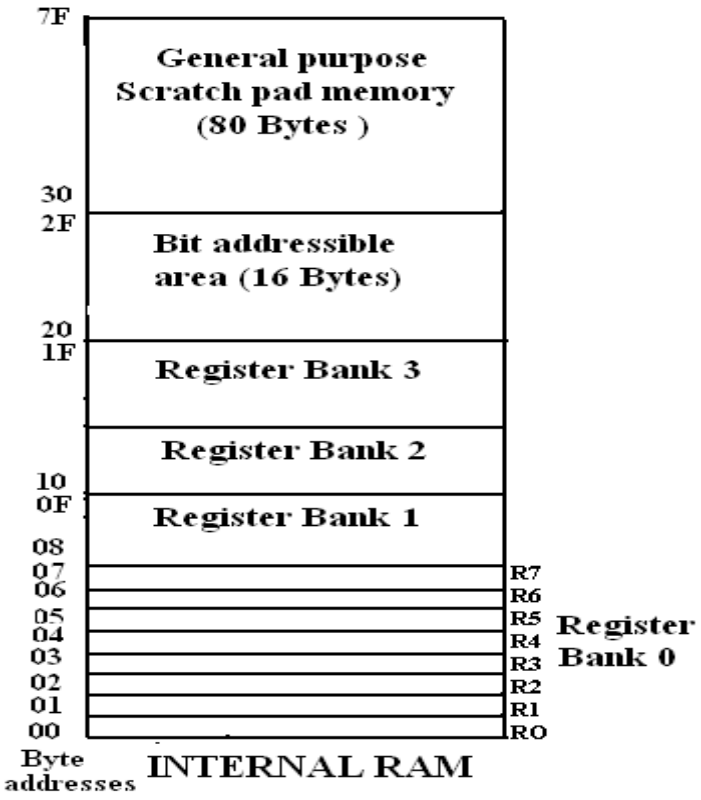

Figure 3.3.1 Internal memory8051 Microcontroller

[Source: "Microprocessor Architecture Programming and Application" by R.S. Gaonkar, page-]

Internal ROM (On –chip ROM):

The 8051 microcontroller has 4kB of on chip ROM but it can be extended up to 64kB.This ROM is also called program memory or code memory. The CODE segment is accessed using the program counter (PC) for opcode fetches and by DPTR for data. The external ROM is accessed when the EA(active low) pin is connected to ground or the contents of program counter exceeds 0FFFH.When the Internal ROM address is exceeded the 8051 automatically fetches the code bytes from the external program memory.

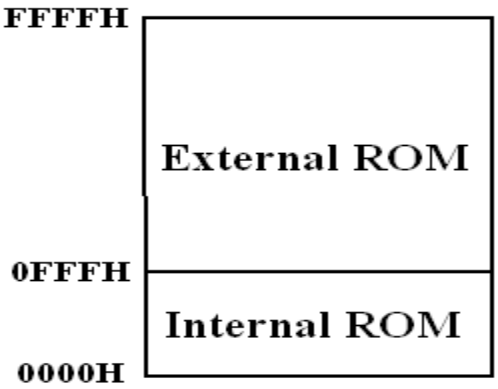

Figure 3.3.2 External memory8051 Microcontroller

[Source: "Microprocessor Architecture Programming and Application" by R.S. Gaonkar, page-]

3.2 PINOUTS

The 8051 microcontroller is available in 40 pin DIP package and it operates on +5 volts DC. Among the 40 pins a total of 32 pins are allotted for the four parallel ports P0, P1, P2 and P3 since each port occupies 8 pins. The remaining 8 pins are VCC, GND, XTAL1, XTAL2, RST, EA, PSEN.

Port 0 occupies a total of 8 pins (pins 1 through 8). It can be used as input port or output port. To configure it as an input port, port bits must be set i.e a high bit must be sent to the port pins. This is normally done by the instruction "SETB".

XTAL1, XTAL2:

These two pins are connected to Quartz crystal oscillator which runs the on-chip oscillator. The quartz crystal oscillator is connected to the two pins along with a capacitor of 30pF as shown in the circuit. If we use a source other than the crystal oscillator, it will be connected to XTAL1 and XTAL2 is left unconnected.

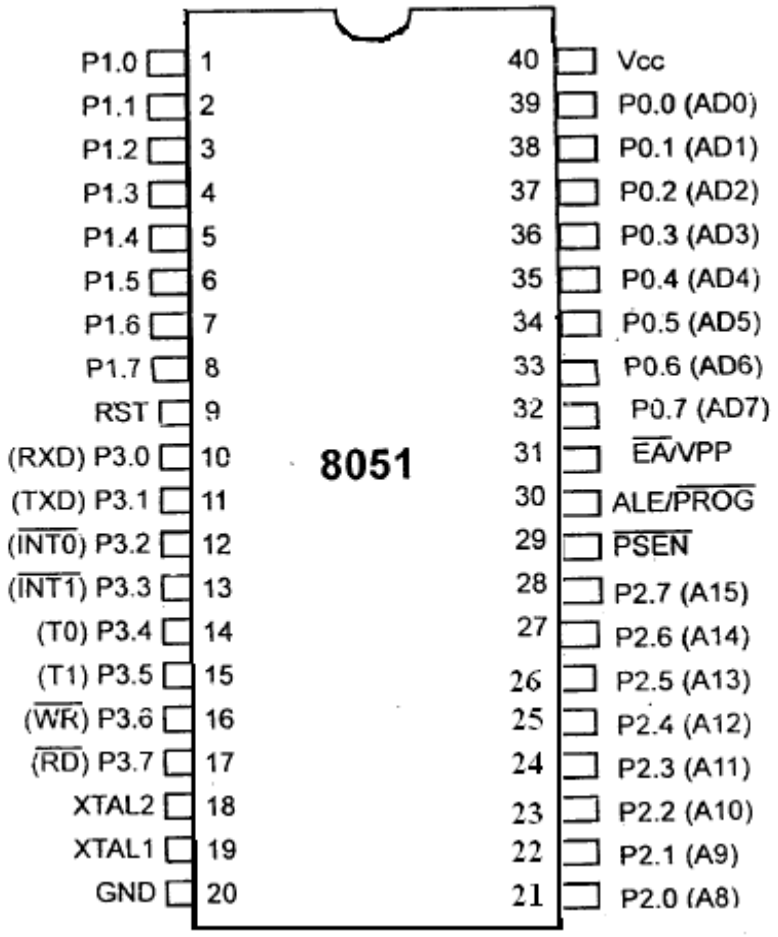

Figure 3.2.1 Pin Diagram of 8051 Microcontroller

[Source: "Microprocessor Architecture Programming and Application" by R.S. Gaonkar, page-]

RST:

The RESET pin is an input pin and it is an active high pin. When a high pulse is applied to this pin the microcontroller will reset and terminate all activities. Upon reset

all the registers except PC will reset to 0000 Value and PC register will reset to 0007 value.

EA (External Access):

This pin is an active low pin. This pin is connected to ground when microcontroller is accessing the program code stored in the external memory and connected to Vcc when it is accessing the program code in the on chip memory. This pin should not be left unconnected.

PSEN (Program Store Enable) :

This is an output pin which is active low. When the microcontroller is accessing the program code stored in the external ROM ,this pin is connected to the OE (Output Enable) pin of the ROM.

ALE (Address latch enable):

This is an output pin, which is active high. When connected to external memory , port 0 provides both address and data i.e address and data are multiplexed through port 0 .This ALE pin will demultiplex the address and data bus .When the pin is High , the AD bus will act as address bus otherwise the AD bus will act as Data bus.

P0.0- P0.7(AD0-AD7) :

The port 0 pins multiplexed with Address/data pins .If the microcontroller is accessing external memory these pins will act as address/data pins otherwise they are used for Port 0 pins.

P2.0- P2.7(A8-A15) :

The port2 pins are multiplexed with the higher order address pins .When the microcontroller is accessing external memory these pins provide the higher order address byte otherwise they act as Port 2 pins.

P1.0- P1.7 :

These 8-pins are dedicated for Port1 to perform input or output port operations.

P3.0- P3.7 :

These 8-pins are meant for Port3 operations and also for some control operations like Read,Write,Timer0,Timer1 ,INT0,INT1 ,RxD and TxD