UNIT IV
MICROCONTROLLER

Architecture of 8051 – Special Function Registers(SFRs) - I/O Pins Ports and Circuits - Instruction set- Addressing modes - Assembly language programming.

4.1 ARCHITECTURE OF 8051

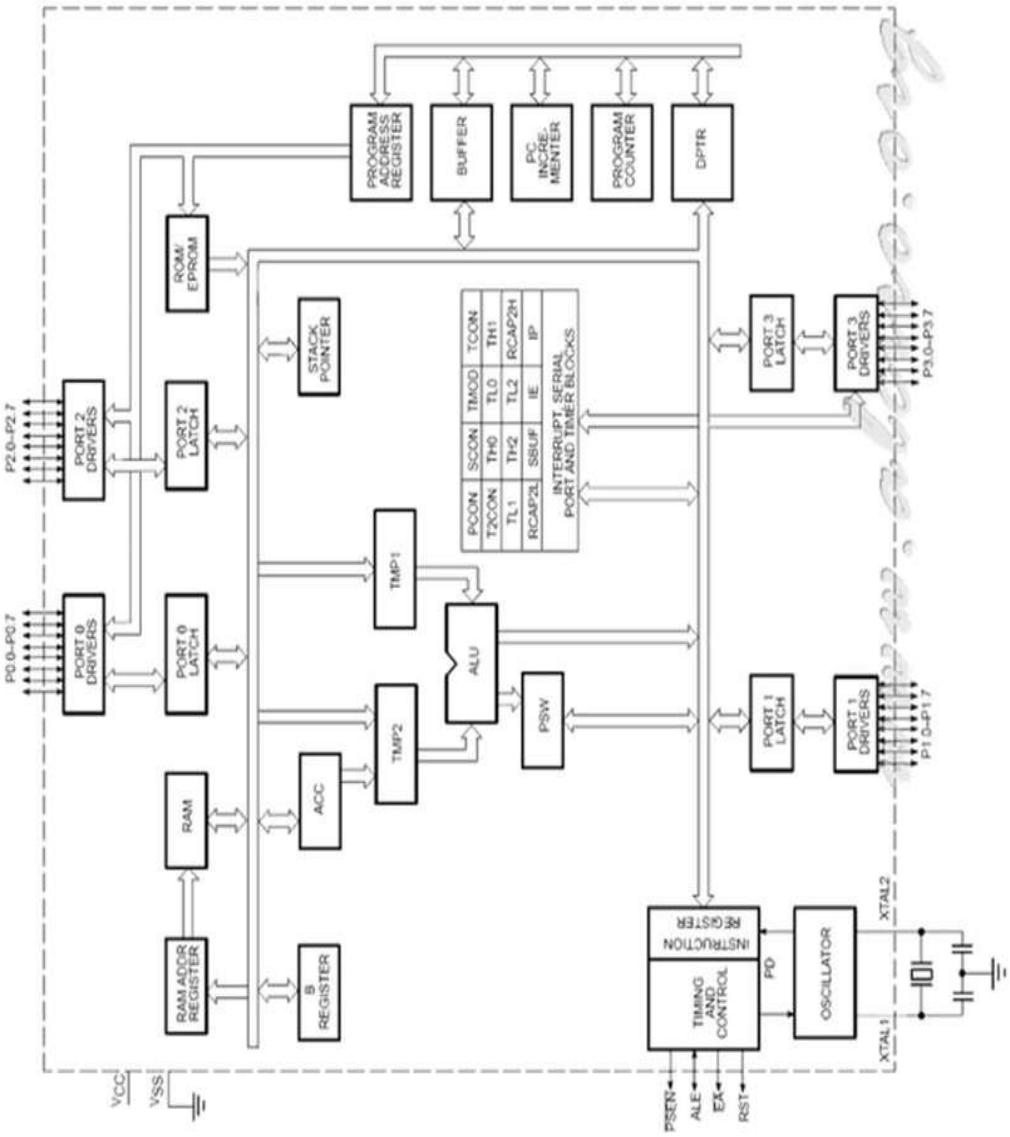


Fig 4.1 Architecture of 8051

The resources offered by 8051 microcontroller are:

1. 8-bit CPU
2. On-chip oscillator
3. 4KB of ROM (Program memory)
4. 128 bytes of RAM (Data memory)
5. 21 Special function registers
6. 32 I/O lines (Ports P0 to P3)
7. 64 KB address space for external data memory
8. 64 KB address space for program memory
9. Two 16-bit timer/counter
10. Five source interrupt structure
11. Full duplex serial port
12. Bit addressability
13. Powerful bit processing capability

Accumulator:

Accumulator is also called ACC or A register. There are many instructions that use the accumulator as the destination. Nearly all arithmetic operations use ACC as one of the operands and after the operation, the result is also stored in ACC.

B Register:

This register is mainly used for multiply and divide operations in which this register acts as one of the operands and after the operations a part of the result is stored in this register. This register otherwise is just like a scratch pad, i.e. a temporary register.

Stack Pointer:

The stack pointer in the 8051 is an 8-bit wide register. This pointer can point to any location in the internal data RAM, i.e. locations from 0-127. When the chip is reset, this register is in initialized to 07H. During PUSH and CALL instructions the stack pointer is first incremented and then the data is stored in the stack.

Data pointer:

The data pointer (DPTR) is a 16-bit register, consisting of a high byte (DPH) and a low byte (DPL). This register normally contains a 16-bit address.

Ports 0-3:

There are 4 bidirectional input/output ports of 8 bits each. These are directly represented as pins of the 8051 chips and are bit addressable. These are multifunctional resources that can be programmed based on the application needs. For example

- P0 and P2 can be used as I/O ports or address lines for external memory.
- P1 can be used as I/O port. It plays an important role in programming of internal memory of 8051.
- P3 can be used as I/O port. Its pins have other important functions like:

  P3.0 Serial Input Line
  P3.1 Serial Output line
  P3.2 External Interrupt line (INT0)
  P3.3 External Interrupt line (INT1)

P3.4 Timer 0 external input (T0)
P3.5 Timer 1 external input (T1)
P3.6 External data memory write strobe (WR)
P3.7 External data memory read strobe (RD)

The bidirectional ports P0 to P3 have the basic structure containing drivers and latches. The port drivers are connected to I/O ports, the latches are connected to the internal bus of the microcontroller.

Serial Data Buffer (SBUF):

This register holds the data that has to be transmitted through the serial port and also holds the data that is received. This register is interconnected to two 8-bit shift registers. When the data is written into the SBUF, it is loaded in the transmit shift register and hence the process of moving a data byte into the SBUF starts the transmission process. During reception, the data coming to the 8051 is clocked into the receive shift register and once all the 8-bits of data or a frame is received, it is transferred to the SBUF.

Control and Status Registers:

All the special function registers (like IP, IE, TMOD, TCON, PCON, SCON, etc.) that are used for controlling the internal resources or to see the status of these resources. These registers contain the control and status bits of the interrupt systems, timers and serial port operations.

Timing and control circuit:

Instruction decoding and control signal generation are performed using the Timing and Control circuit block.

Memory Organization:

8051 contains 128 bytes RAM as data memory and 4KB ROM as program memory on the chip. 8051 can access up to 64 KB of external data memory and also the internal data RAM locations.

Program memory:

The 64 KB of program memory also includes the 4KB of the on-chip ROM. The processor will come to know whether the user wants to use the internal ROM or not from the status of the $\overline{EA}$ pin. If this pin is pulled low, it means that the user does not want to use the internal ROM available. Hence, the processor will access 0000-FFFFH from the external program memory. If this pin is held high, the processor will access addresses 0000-0FFFH from the internal ROM and as the address goes above 0FFFH, it will access the external program memory that is interfaced with it. In short,

If $\overline{EA}$ = 0 V
External program memory = 0000-FFFFH (64K)
If $\overline{EA}$ = 5 V
Internal program memory (ROM) = 0000-0FFFH (4K)
External program memory = 1000-FFFFH (60K)

Data memory:

The 8051 can access 64 KB of the external data memory and 128 bytes of the internal data RAM and also 21 special function registers.

For accessing the external data memory, the processor can either issue an 8-bit address or a 16-bit address. To access the internal data memory, the 8-bit address is used. This 8-bit address can provide address spaces for 256 locations. The lower 128 addresses (0 to 127) are used as 128 bytes on-chip RAM and the upper part of the address space, i.e.(128 to 255) is used to address the various special function registers.

The lowest 32 bytes (0-31) are reserved for 4 banks of 8 registers, each R0-R7, out of which one bank may be used at any time. The working bank is specified in two bits of the Program Status Word Register.

Program Status Word

7	6	5	4	3	2	1	0
CY	AC	FO	RS1	RS0	OV	X	P

CY - Carry flag
AC - Auxiliary carry flag
FO - Overflow flag
OV - Odd parity flag
P - User flag 0

RS1	RS0	Register Bank
0	0	Register Bank 0
0	1	Register Bank 1
1	0	Register Bank 2
1	1	Register Bank 3

4.1.1 Internal architecture:

Internal data RAM with address register is connected to the internal bus. The address register will get the address from the internal bus and the data will be transferred to the specified register through the bus.

In order to access the internal program memory (ROM) as well as the external program memory, the program address register is interfaced to the program counter which, in turn, is connected to the incrementer circuit for incrementing the PC.

A bidirectional buffer has been provided to temporarily store the branch address as well as the operand address information. This will also facilitate the transfer of the PC contents to other circuits to calculate the PC relative jump address.

Ports P0 and P2 also facilitate the connection to the external memory. The lower-order address bits (A_0 to A_7) are sent through port 0 (P0.0 to P0.7) and the higher-order address bits (A_8 to A_{15}) are sent through port 2 (P2.0 to P2.7). P0 also functions as data bus during external read/write in a time multiplexed manner. Data pointer register is interfaced to the address register.

Two temporary registers TMP1 and TMP2 are connected to ALU. These registers hold the operands for calculation. The PSW register is directly interface3d to ALU for updation on the basis of last operation.

4.1.2 Pins and signals:

Vcc and Vss: Vcc is the power supply pin. It is connected to a 5V regulated supply. Vss is the ground pin.

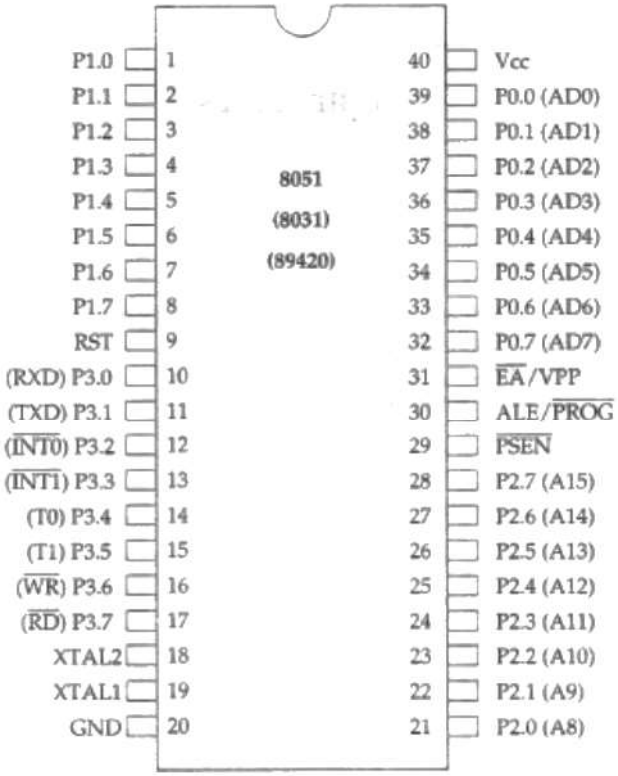


Fig 4.2: Pin diagram of 8051

Port 0 (P0.0 to P0.7): These are 8-bit bidirectional input/output port pins. They are bit addressable. They can sink up to eight LS TTL logic gates. These pins also act as the lower part of the address bus as well as data bus while accessing the external memory. The lower part of the address bus is also time multiplexed with the data bus.

Port 1 (P1.0 to P1.7): Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. That means an open collector output can be directly connected to this port without an external pull-up. It plays an important role during the programming of the internal memory. This port can sink/source three LS TTL inputs.

Port 2 (P2.0 to P2.7): This is also an 8-bit bidirectional I/O with internal pull-ups. This also has an alternate function of higher-order address byte, while accessing the external memory. It plays an important role during the programming of the internal ROM.

Port 3 (P3.0 to P3.7): This is also an 8-bit bidirectional I/O with internal pull-ups. This also serves many other important alternate functions which enable the 8051 to be called a microcontroller.

ALE/$\overline{PROG}$: Address Latch Enable, also known as ALE, is an output pin and is used for latching the lower-order lines A_0 to A_7 of the address bus as the lines are also time multiplexed with data lines. A pulse is generated at a constant rate of 1/6 oscillator frequency. This pulse is always generated except in few cases like external data memory access, where one ALE pulse is skipped during each access. This pin also acts as an input pin for the programming pulses during the programming of the internal EPROM.

XTAL1 and XTAL2: The 8051 has an internal clock circuit, hence a crystal of proper frequency can be connected directly to these two pins. The frequency range is 3.5-12 MHz. An external oscillator can also be connected instead of a crystal. In this case, the XTAL2 pin is grounded and the oscillator signal is given to the XTAL2 pin.

$\overline{PSEN}$**:**This pin gives out active low pulses.This signal is used for fetching the data from the external program memory. A pulse is generated after every six clock cycles. This is used as a read signal for reading from external program memory. If the data to be fetched is inside the chip itself, then $\overline{PSEN}$ is not generated. This signal is also generated during the external data memory operation.

$\overline{EA}$**/VPP:** This pin, is connected to 5V, will indicate to the processor that the 4KB of the program memory within the chip should be used. Hence address 0000H to 0FFFH will be within the chip. Once the address exceeds this, i.e. 1000H to FFFFH, the external program memory is accessed. This pin, if pulled low, will inform the processor that the on-chip ROM is not being used. Hence the entire 64KB will be external to the chip. So, depending on whether a designer is using the internal program memory or not.

RST/VPD:A high on this pin, for more than 24 oscillator cycles, will reset the chip. VPD may be used to supply power to the internal RAM during power failure or power down modes.

4.2 BIT ADDRESSES FOR I/O AND RAM

Many microprocessors such as the 386 or Pentium allow programs to access registers and I/O ports in byte size only. In other words, if you need to check a single bit of an I/ O port, you must read the entire byte first and then manipulate the whole byte with some logic instructions to get hold of the desired single bit. This is not the case with the 8051. Indeed, one of the most important features of the 8051 is the ability to access the registers, RAM, and I/ O ports in bits instead of bytes. This is a very unique and powerful feature for a microprocessor made in the early 1980s.

4.2.1 Bit-addressable RAM

Of the 128-byte internal RAM of the 8051, only 16 bytes are bit-addressable. The rest must be accessed in byte format. The bit addressable RAM locations are 20H to 2FH. These 16 bytes provide 128 bits of RAM bit-addressability since 16 x 8 = 128. They are addressed as 0 to 127 (in decimal) or 00 to 7FH. Therefore, the bit addresses 0 to 7 are for the first byte of internal RAM location 20H, and 8 to 0FH are the bit addresses of the second byte of RAM location 21H, and so on. The last byte of 2FH has bit addresses of 78H to 7FH. See Figure 4.3.

Note that internal RAM locations 20 - 2FH are both byte-addressable and bit-addressable. In order to avoid confusion regarding the addresses 00 - 7FH, the following two points must be noted.

1. The 128 bytes of RAM have the byte addresses of 00 - 7FH and can be accessed in byte size using various addressing modes such as direct and register-indirect, as we have seen in this chapter and previous chapters. These 128 bytes are accessed using byte-type instructions.
2. The 16bytes of RAM locations 20 - 2FH also have bit addresses of 00 - 7FH since 16 x 8 = 128 (00 - 7FH). In order toaccess these 128bits of RAMlocations and other bit-addressable space of 8051individually, wecan use only the single-bit instructions such as SETB. Notice that the single-bit instructions use only one addressing mode and that is direct addressing mode. It must be noted that there is no indirect addressing mode for single-bit instructions.

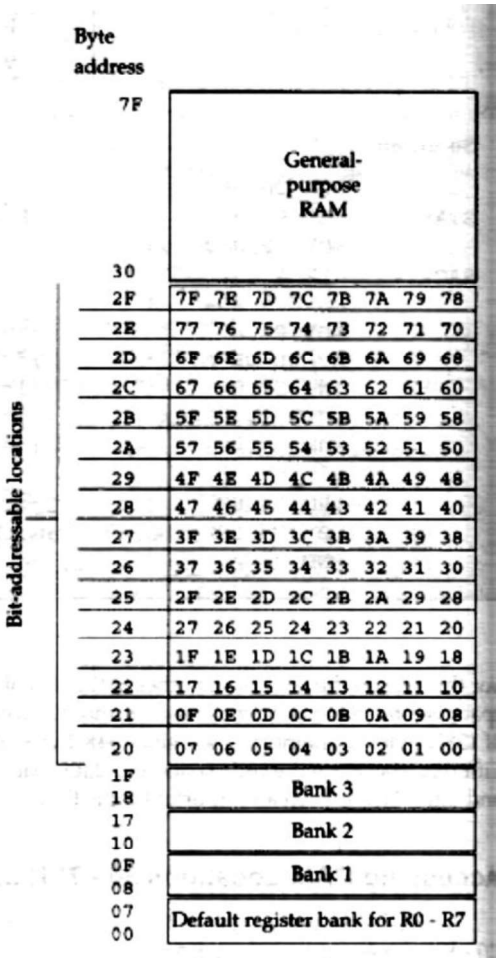


Fig 4.3: 16 Bytes of Internal RAM.

4.2.2 I/O port bit addresses

8051 has four 8-bit I/O ports: P0,Pl, P2, and P3. We can access either the entire 8 bits or any single bit without altering the rest. When accessing a port in a single-bit manner, we use the syntax "SETB where X is the port number 0, 1, 2, or 3, and Y is the desired bit number from

0 to 7 for data bits D0 to D7. See fig 4.4. For example, "SETB Pl.5" sets high bit 5 of port 1. Remember that D0 is the LSBand D7 is the MSB.

Every SFR register is assigned a byte address and ports P0- P3 are part of the SFR. For example, P0 is assigned byte address 80H, and Pl has address of 90H as shown in Figure 4.4. While all of the SFR registers are byte-addressable some of them are also bit-addressable. The P0- P3 are among this category of SFR registers. From Figure 4.4 we see.that the bit addresses for P0 are 80H to 87H, and for Pl are 90H to 97H, and so on.

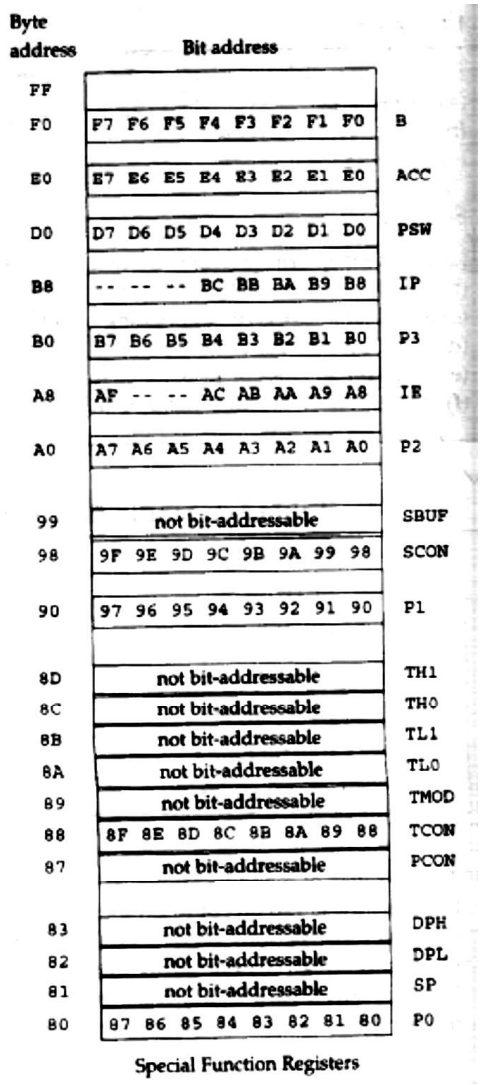


Fig 4.4: SFR RAM Address(Byte and Bit)

4.2.3 Bit memory map

From fig 4.3, 4.4 and fig 4.5 notice the following facts.

1. The bit addresses 00 - 7FH are assigned to RAM locations of 20 - 2FH.

2. The bit addresses 80 - 87H are assigned to the P0 port.
3. The bit addresses 88 - 8FH are assigned to the TCON register.
4. The bit addresses 90 - 97H are assigned to the Pl port.
5. The bit addresses 98 - 9FH are assigned to the SCON register.
6. The bit addresses A0- A7H are assigned to the P2 port.
7. The bit addresses A8-AFH are assigned to the IE register.
8. The bit addresses B0- B7H are assigned to the P3 port.
9. The bit addresses B8- BFH are assigned to IP.
10. The bit addresses C0- CFH are not assigned.
11. The bit addresses D0- D7H are assigned to the PSW register.
12. The bit addresses D8 - DFH are not assigned.
13. The bit addresses E0- E7H are assigned to the Accumulator register.
14. The bit addresses E8 - EFH are not assigned.
15. The bit addresses F0- F7H are assigned to the B register.

P0	Addr	P1	Addr	P2	Addr	P3	Addr	ports
P0.0	80	P1.0	90	P2.0	A0	P3.0	B0	D0
P0.1	81	P1.1	91	P2.1	A1	P3.1	B1	D1
P0.2	82	P1.2	92	P2.2	A2	P3.2	B2	D2
P0.3	83	P1.3	93	P2.3	A3	P3.3	B3	D3
P0.4	84	P1.4	94	P2.4	A4	P3.4	B4	D4
P0.5	85	P1.5	95	P2.5	A5	P3.5	B5	D5
P0.6	86	P1.6	96	P2.6	A6	P3.6	B6	D6
P0.7	87	P1.7	97	P2.7	A7	P3.7	B7	D7

Fig 4.5:Bit addresses for all ports

4.2.4 Registers bit-addressability

CY	AC	F0	RS1	RS0	OV	–	P

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

Fig 4.6: Bits of PSW register

While all I/O ports are bit-addressable, that is not the case with registers, as seen from Figure 4.3. Only registers A, B, PSW, IP, IE, ACC, SCON, and TCON are bit-addressable. Of the bit-addressable registers, we will concentrate on the familiar registers A, B, and PSW. The Now let's see how we can use bit-addressability of registers such as A and PSW. The PSW register two bits are set aside for the selection of the register banks. See Figure 4.6.Upon RESET,bank 0 is selected. We can select any other banks using the bit-addressability of the PSW. The bit addressability of PSW also eliminates the need for instructions such as JOV (Jump if OV=l).

Using BIT directive :The BIT directive is a widely used directive to assign the bit-addressable I/O and RAM locations. The BIT directive allows a program to assign the I/ O or RAM bit at the beginning of the program, making it easier to modify them.
Using EQU directive: We can also use the EQU directive to assign addresses.

4.3 SPECIAL FUNCTION REGISTERS

The resources in 8051 can be accessed through special function registers maintained in the internal data memory. Resources like timers require setting the modes and controlling them. Similarly, serial data input would require a serial data buffer as well as the control. 8051 offers 4 ports which can be used for various purposes like input/output, memory interface, etc. The function of these resources can be programmed by special function registers. (See Fig 4.4)

These registers have special function like controlling the timer/counter, enabling interrupts, controlling the serial port operations, etc. There are 21 special function registers in 8051. Some special registers are bit addressable. Various special function registers are:

ACC	Accumulator	0E0H
B	B register	0F0H
PSW	Program status word	0D0H
SP	Stack Pointer	81H
DPTR	Data pointer 2 bytes	
DPTR(Low)	Data pointer low	82H
DPTR(High)	Data pointer high	83H
P0	Port 0	80H
P1	Port 1	90H
P2	Port 2	0A0H
P3	Port 3	0B0H
IP	Interrupt Priority Control	0B8H
IE	Interrupt Enable Control	0A8H
TMOD	Timer/counter mode	89H
TCON	Timer/counter control	88H
T2CON	Timer/counter 2 control	0C8H
T2MOD	Timer/counter 2 mode	0C9H
TH0	(Timer/Counter) 0 High	8CH
TL0	(Timer/Counter) 0 Low	8AH
TH1	(Timer/Counter) 1 High	8DH
TL1	(Timer/Counter) 1 Low	8BH
RCAP2H	T/C2 capture register high byte	0CBH
RCAP2L	T/C2 capture register low byte	0CAH
SCON	Serial control	98
SBUF	Serial Data Buffer	99
PCON	Power Control	87

The following two points should be noted about the SFR addresses.

1. The special function registers have addresses between 80H and FFH. These addresses are above 80H, since the addresses 00 to 7FH are addresses of RAM memory inside the 8051.
2. Not all the address space of 80 to FF is used by the SFR. The unused locations 80H to FFH are reserved and must not be used by the 8051 programmer.

Regarding direct addressing mode, notice the following two points:

(a) the address value is limited to one byte, 00 - FFH, which means this addressing mode is limited to accessing RAM locations and registers located inside the 8051.

(b) if you examine the 1stfile for an Assembly language program, you will see that the SFR registers.

Accumulator:

Accumulator is also called ACC or A register. There are many instructions that use the accumulator as the destination. Nearly all arithmetic operations use ACC as one of the operands and after the operation, the result is also stored in ACC.

B Register:

This register is mainly used for multiply and divide operations in which this register acts as one of the operands and after the operations a part of the result is stored in this register. This register otherwise is just like a scratch pad, i.e. a temporary register.

Program Status Word

7	6	5	4	3	2	1	0
CY	AC	FO	RS1	RS0	OV	X	P

CY - Carry flag
AC - Auxiliary carry flag
FO - Overflow flag
OV - Odd parity flag
P - User flag 0

RS1	RS0	Register Bank
0	0	Register Bank 0
0	1	Register Bank 1
1	0	Register Bank 2
1	1	Register Bank 3

Stack Pointer:

The stack pointer in the 8051 is an 8-bit wide register. This pointer can point to any location in the internal data RAM, i.e. locations from 0-127. When the chip is reset, this register is in initialized to 07H. During PUSH and CALL instructions the stack pointer is first incremented and then the data is stored in the stack.

Data pointer:

The data pointer (DPTR) is a 16-bit register, consisting of a high byte (DPH) and a low byte (DPL). This register normally contains a 16-bit address.

Ports 0-3:

There are 4 bidirectional input/output ports of 8 bits each. These are directly represented as pins of the 8051 chips and are bit addressable. These are multifunctional resources that can be programmed based on the application needs. For example

- P0 and P2 can be used as I/O ports or address lines for external memory.
- P1 can be used as I/O port. It plays an important role in programming of internal memory of 8051.
- P3 can be used as I/O port. Its pins have other important functions like:

  P3.0 Serial Input Line
  P3.1 Serial Output line
  P3.2 External Interrupt line (INT0)
  P3.3 External Interrupt line (INT1)
  P3.4 Timer 0 external input (T0)
  P3.5 Timer 1 external input (T1)
  P3.6 External data memory write strobe (WR)
  P3.7 External data memory read strobe (RD)

  The bidirectional ports P0 to P3 have the basic structure containing drivers and latches. The port drivers are connected to I/O ports, the latches are connected to the internal bus of the microcontroller.

Interrupt Priority (IP) Register

D7							D0
-	-	-	PS	PT1	PX1	PT0	PX0

PS Serial Port Interrupt Priority level
PT1 Timer 1 Interrupt Priority level
PX1 External Interrupt 1 Priority level
PT0 Timer 0 Interrupt Priority level
PX0 External Interrupt 0 Priority level
Priority bit=1 assigns high priority: priority bit =0 assigns low priority

Interrupt Enable (IE) Control

D7							D0
EA	X	X	ES	ET1	EX0	ET0	EX0

EA (enable all):

If EA=0, no interrupt will be acknowledged. If EA=1, each interrupt source is individually enabled or disabled by setting or clearing the enable bit.

ES Enables or disables Serial port interrupt:

If ES=0, serial port interrupt is disabled.

ET1 Enables or disables the Timer 1 overflow interrupt:

If ET1 = 0 Timer 1 overflow interrupt is disabled

EX1 Enables or disables the external interrupt 1

If EX1 =0 external interrupt 1 is disabled

ET0 Enables or disables Timer 0 overflow interrupt

If ET0 = 0 Timer 0 overflow interrupt is disabled

EX0 Enables or disables the external interrupt 0

If ET0 = 0 external interrupt 0 is disabled

Timer/counter mode (TMOD)

Both timers 0 and 1 use the same register, called TMOD, to set the various timer operation modes, TMOD is an 8 bit register in which the lower 4 bits are set aside for Timer 0 and the upper 4 bits for Timer 1. In each case, the lower 2 bits are used to set the timer a mode and the upper 2 bits to specify the operation.

(MSB) (LSB)

GATE	C/$\overline{T}$	M1	M0	GATE	C/$\overline{T}$	M1	M0
Timer1				Timer 0			

GATE:

When GATE = 0, the start and stop of timer are controlled by software by means of TR (timer start) bits TR0 and TR1. This is achieved by the instructions "SETB TR1" and "CLR TR1" for Timer 1, and "SETB TR0" and CLR instruction.

When GATE = 1, the start and stop of timer are controlled by hardware.

C/$\overline{T}$:

This bit in the TMOD register is used to decide whether the timer is used as a delay generator or an event counter. If C/$\overline{T}$ = 0, it is used as a timer for time the 8051. If C/$\overline{T}$ = 1, it is used as a counter.

M1,M0 :

M1- Mode bit 0, M2- Mode bit 0

M1	M0	Mode	Operating Mode
0	0	1	13 bit timer mode
0	1	1	16 bit timer mode
1	0	2	8 bit timer with auto reload
1	1	3	Split timer mode

Timer/counter control (TCON)

TCON register

TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

TF1 : Timer 1 overflow flag.

Set by hardware when the timer/counter overflows.

Cleared by hardware when the processor vectors to the interrupt routine.

TR1 : Timer 1 run control bit.

Set/cleared by software to turn the timer/counter on/off.

TF0 : Timer 0 overflow flag.

Set by hardware when the timer/counter overflows.

Cleared by hardware when the processor vectors to the interrupt routine.

TR0 : Timer 0 run control bit.

Set/cleared by software to turn the timer/counter on/off.

IE1 : Interrupt 1 edge flag.

Set by hardware when the external interrupt edge is detected.

Cleared when the interrupt is processed.

IT1 : Interrupt 1 type control bit

Set/ Cleared by software to specify the falling edge/low level triggered external interrupts.

- When IT1 = 1 then $\overline{INT1}$ is falling edge triggered
- When IT0 = 0 then $\overline{INT1}$ is low-level triggered

IE0 : Interrupt 0 edge flag.

Set by hardware when the external interrupt edge is detected.

Cleared when the interrupt is processed.

IT0 : Interrupt 0 type control bit

Set/ Cleared by software to specify the falling edge/low level triggered external interrupts.

- When IT0 = 1 then $\overline{INT0}$ is falling edge triggered
- When IT0 = 0 then $\overline{INT0}$ is low-level triggered

(Timer/Counter) 0 High, (Timer/Counter) 0 Low

The 16-bit register of Timer 0 is accessed as low byte and high byte. The low byte register is called TL0 (Timer 0 low byte) and the high byte register is referred to as TH0 (Timer 0 high byte).

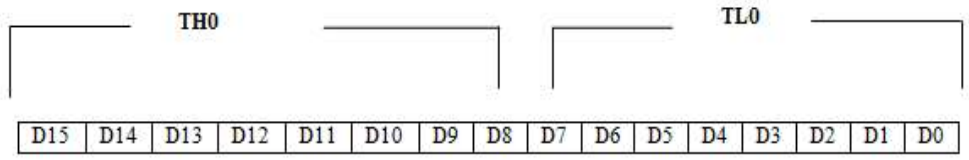
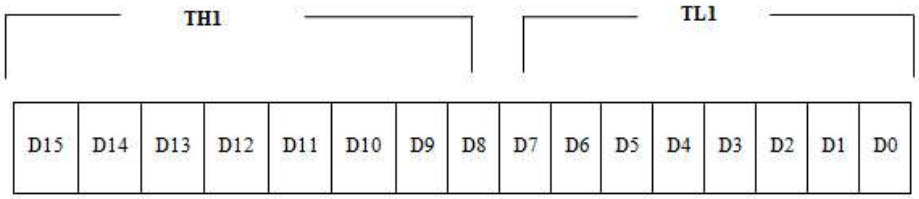


Fig 4.7:Timer 0 registers

Timer/Counter) 1 High, (Timer/Counter) 1 Low

Timer 1 is also 16 bits, and its 16 bit register is split into two bytes, referred to as TL1



(Timer 1 low byte) and TH1 (Timer 1 high byte).

Fig 4.8: Timer 1 registers

SCON (serial control) register

The SCON register is an 8-bit register used to program the start bit, stop bit, and data bits of data framing, among other things.

SM0	SM1	SM2	REN	TB8	RB8	T1	RI

Fig 4.9: SCON Serial Port Control Register (Bit-Addressable)

SM0, SM1

SM0 and SM1 are D7 and D6 of the SCON register, respectively. These two bits determine the framing of data by specifying the number of bits per character, and the start and stop bits. They take the following combinations.

SM0	SM1	Mode	Baud Rate
0	0	Serial mode 0 (8 data bits)	1/12th of the oscillator frequency
0	1	Serial mode 1(1 bit for start, 8 bits of data, 1 bit for stop)	Baud rate is variable, depends on timer 1 overflow rate.
1	0	Serial mode 3(1 bit for start, 8 bits of data, 1 bit can be programmed, 1 bit for stop)	1/32 or 1/64th of the oscillator frequency.
1	1	Serial mode 3(1 bit for start, 8 bits of data, 1 bit can be programmed, 1 bit for stop)	Baud rate is variable, depends on timer 1 overflow rate.

SM2

SM2 is the D5 bit of the SCON register. This bit enables the multiprocessing capability of the 8051.

REN

The REN (receive enable), bit is D4 of the SCON register. When the REN bit is high, it allows the 8051 to receive data on the RxD pin of the 8051. As a result if we want the 8051 to both transfer and receive data, REN must be set to 1. By making REN = 0, the receiver is disabled.

TB8

TB8 (transfer bit 8) is bit D3 of SCON. It is used for serial modes 2 and 3. We make TB8 = 0 since it is not used in our applications.

RB8

RB8 (receive bit 8) is bit D2 of the SCON register. In serial mode 1, this bit gets a copy of the stop bit when an 8-bit data is received.

TI

TI (transmit interrupt) is bit Dl of the SCON register. This is an extremely important flag bit in the SCON register. When the 8051 finishes the transfer of the 8-bit character, it raises the TI flag to indicate that it is ready to transfer another byte. The TI bit is raised at the beginning of the stop bit.

Rl

RI (receive interrupt) is the DO bit of the SCON register. This is another extremely important flag bit in the SCON register. When the 8051 receives data serially via RxD, it gets rid of the start and stop bits and places the byte in the SBUF register. Then it raises the RI flag

bit to indicate that a byte has been received and should be picked up before it is lost. RI is raised halfway through the stop bit.

Serial Data Buffer

This register holds the data that has to be transmitted through the serial port and also holds the data that is received. This register is interconnected to two 8-bit shift registers. When the data is written into the SBUF, it is loaded in the transmit shift register and hence the process of moving a data byte into the SBUF starts the transmission process. During reception, the data coming to the 8051 is clocked into the receive shift register and once all the 8-bits of data or a frame is received, it is transferred to the SBUF.

Power Control register (PCON):

D7							D0
SMOD	-	-	-	GF1	GF0	PD	IDL

Option 1 is not feasible in many situations since the system crystal is fixed. Therefore, we will explore option 2. There is a software way to double the baud rate of the 8051 while the crystal frequency is fixed. This is done with the register called PCON (power control). The PCON register is an 8-bit register. Of the 8 bits, some are unused, and some are used for the power control capability of the 8051. The bit that is used for the serial communication is D7, the SMOD (serial mode) bit. When the 8051 is powered up, D7 (SMOD bit) of the PCON register is zero. We can set it to high by software and thereby double the baud rate.

4.4 I/O PINS PORTS AND CIRCUITS

4.4.1 8051 I/O programming

In 8051 there are a total of four ports for I/O operations. Of the 40 pins, a total of 32 pins are set aside for the four ports P0, Pl, P2, and P3, where each port takes 8 pins. The rest of the pins are designated as Vcc, GND, XTALl, XTAL2,RST,$\overline{EA}$, ALE/$\overline{PROG}$ and $\overline{PSEN}$.

4.4.2. I/O port pins and their functions

The four ports P0,Pl, P2, and P3 each use 8 pins, making them 8-bit ports. All the ports upon RESET are configured as inputs, ready to be used as input ports. When the first 0 is written to a port, it becomes an output. To reconfigure it as an input, a 1 must be sent to the port. To use any of these ports as an input port, it must be programmed.

4.4.3 Port 0

Port 0 is an 8-bit open collector bidirectional I/O port. It can perform two functions.

- Simple I/O operation
- External memory interface for lower order address bus and data bus.

Simple I/O operation:

When a 1 is loaded to the latch as part of the input operation both the FETs T1 and T2 are turned off. This causes port 0 pin to float in high impedance state and it gets connected to the pin read buffer. An external pull-up register to supply a high output. Thus the port is configured for input operations. Information of P0 pin is transferred to internal bus when the read pin signal is generated.

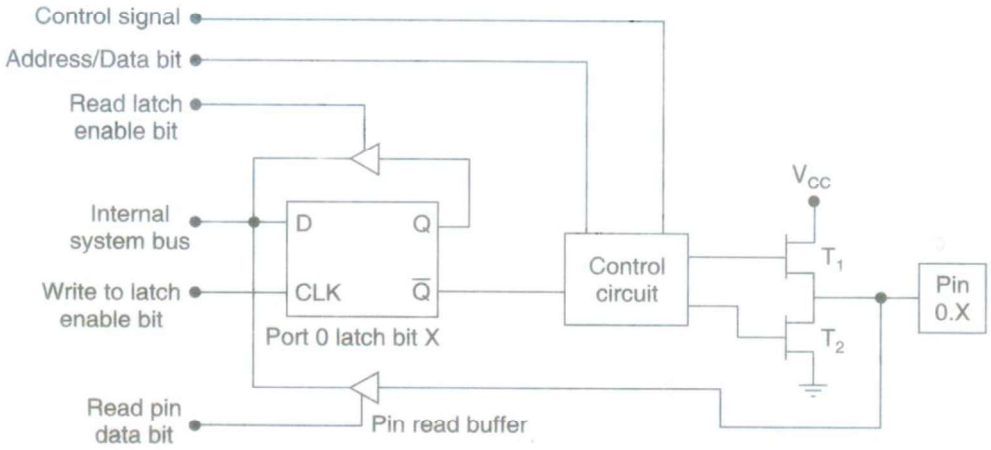


Fig 4.10: Port 0 configuration

When used as an output port, data will be directly loaded to latch. If a 0 is the output on the latch, it will turn on FET T2 and thus the output pin will be grounded. If a 1 is being output, the output pin will be float to high impedance state and the external pull-up register may be used to output 1 as the data state.

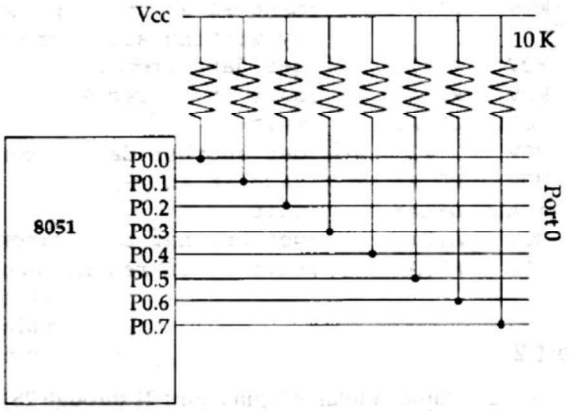


Fig 4.11: Port 0 with pull-up resistors

External Memory Interface:

For any read/write to the external memory, first the address and then the data transfer will be done. When the address is to be transferred to the pin, the control signals will connect the address information directly to the pin through FETs T1 and T2. The port latch is disconnected in this case.

If the address bi t=1, then T1 will be turned ON and T2 will be turned OFF. This will cause logic 1 signal on the port pin. When the address bit =0, then T1 is turned OFF and T2 is turned ON. This will ground the output pin and provide the logic 0 signal.

After the transfer of the address information, if memory read is required the 8051 places a 1 on the latch and thus reconfigures the pin to receive the data.

4.4.4 Port 1

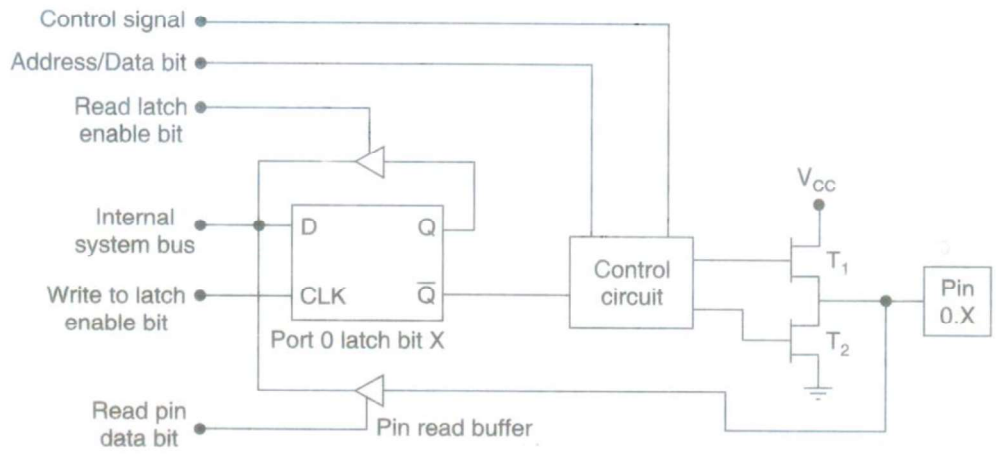


Fig 4.12 : Port 1 configuration

Port 1 is bidirectional I/O port with internal pull-ups. The circuit has three FETs T1, T2 and T3. FETs T2 and T3 work as internal pull-up to T1.

When Port 1 is used as input, logic level 1 is loaded to the latch. This will turn OFF the FET T1. This will effectively float the port pin to high impedance state and will be connected to the pin read buffer. The pin is at logic level 1 at this instant. Any external device may place a 0 by driving the pin to the ground, or it may place a 1 on the pin by leaving it at logic state 1.

When used as an output port, the value will be loaded to the latch. The latches which contain level 1 will turn OFF FET T1 and this will drive the pin high through the pull-up register formed by T2 and T3. The latches containing level 0 will turn on the FET T1. This will ground the port pin and it will show logic level 0.

The FETs T2 and T3, forming internal pull-up, are depletion type and enhancement type. The FET T3 is turned on for two clock pulses which provide low impedance path to the supply voltage. This reduces transients in the external circuit and helps in high-speed operation.

4.4.5 Port 2

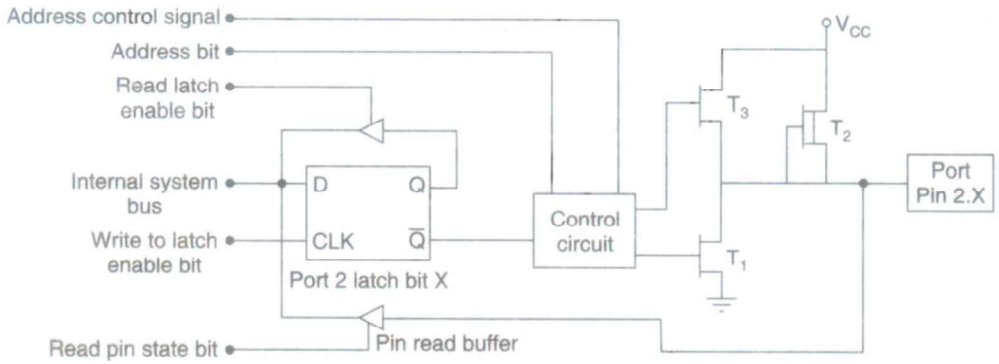


Fig 4.13: Port 2 configuration

Port 2 can be used as a bidirectional I/O port or it can also be used to output higher-order address bus for external memory interface.

When used as an input port, logic level 1 is stored in the latch. The FET T1 is turned ON and floats in high impedance state. The pin is directly connected to pin Read Buffer. The external device can place a 0 or a 1 on the pin.

When used as an output port, the working of Port 2 is similar to port 1. This means the latches containing 0 will turn ON FET T1, thus grounding the pin. The latches containing 1 will turn OFF T1 and thus the pin will be driven to logic level 1.

When port 2 is used to output higher-order address bits for external memory access, the control signal will bypass the latch and the address bits are directly connected to the pin through FETs.

4.4.6 Port 3

Port 3 is a bidirectional I/O port with internal pull-up. Different pins of Port 3 also facilitate alternate functions. The function of port 3 pins is either under the control of the port latches or under the control of the special function registers. These pins are individually programmable unlike other ports where alternate functions of all port pins are programmed together.

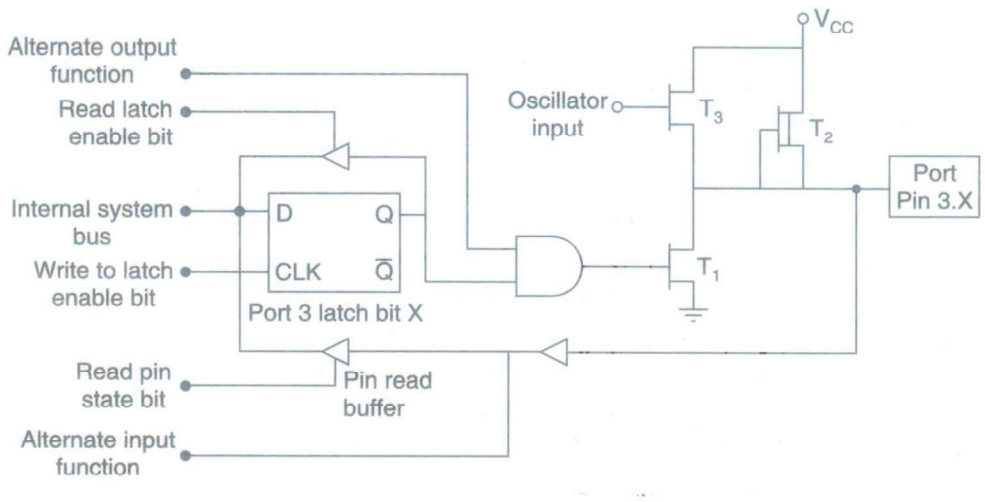


Fig 4.14 : Port 3 configuration

Following are the alternate functions of different pins of port 3.

P3.0 Serial data input (RXD)
P3.1 Serial data output (TXD)
P3.2 External interrupt 0 ($\overline{INT0}$)
P3.3 External interrupt 1 ($\overline{INT1}$)
P3.4 Timer 0 external input (T0)
P3.5 Timer 1 external input (T1)
P3.6 External data memory write strobe ($\overline{WR}$)
P3.7 External data memory read strobe ($\overline{RD}$)

Read-modify-write feature

The instructions that read the port latch normally read a value, perform an operation (and possibly change it), then rewrite it back to the port latch. This is often called *"Read-Modify-Write"*.

The ports in the 8051 can be accessed by the read-modify-write technique. This feature saves many lines of code by combining in a single instruction all three actions of

1. Reading the port
2. Modifying its value and
3. Writing to the port.

4.5 INSTRUCTION SET

4.5.1 Data Transfer Instructions:

Data transfer instructions are divided into three classes:

1. General purpose transfers
2. Accumulator-specific transfers
3. Address-object transfers

1. General Purpose Transfers

Three general purpose transfers are provided, which may be applied to most operands.

MOV <dest byte>, <scr byte>

The byte variable indicated by the second operand is copied into the location specified by the first operand. The source byte is not affected. No other register or flag is affected.

Eg: MOV A, Rn

(A) ← (Rn)

MOV A, Direct

(A) ← (Direct)

MOV A, @ Ri

(A) ← ((Ri))

MOV A, #data

(A) ← (data)

MOV @ Ri, A

((Ri)) ← (A)

PUSH direct (Push onto stack)

The stack pointer is incremented by one. The contents of the indicated variable are then copied into the internal data Ram location addressed by the attack pointer.

(SP) ← (SP) + 1

((SP)) ← (direct)

POP direct (Pop from stack)

The contents of the internal data Ram location addressed by the attack pointer are read and the stack pointer is decremented by one. The value is then transferred directly addressed byte indicated.

(direct) ← ((SP))

(SP) ← (SP) - 1

2. Accumulator specific Transfers:

Four accumulator specific transfers are provided.

XCH A, <byte> (Exchange accumulator with byte variable)

XCH exchanges the contents of accumulator with those indicated variable. The source/destination operand can be register, direct or register-indirect addressing.

XCH A, Rn

(A) ↔ (Rn)

XCH A, Direct

(A) ↔ (Direct)

XCH A, @ Ri

(A) ↔ ((Ri))

XCHD A, @Ri (Exchange digit)

Exchanges the lower nibble of the accumulator (bits 3-0) with that of the internal RAM location indirectly addressed by the specified register.

(A)3-0 ↔ ((Ri))3-0

MOVX <dest byte>, <scr byte>

The MOVX instruction transfers data between the accumulator and a byte of the external data memory.

MOVX A, @ Ri

(A) ← ((Ri)

MOVX @ Ri, A

((Ri) ← (A)

MOVX A, @ DPTR

(A) ← ((DPTR))

MOVX @ DPTR, A

(DPTR) ← (A)

MOVC A, @ A+ <base register>

The MOVC instruction loads the accumulator with a code byte, or a constant from the program memory. No flags are affected.

MOVC A, @ A+ DPTR

(A) ← ((A) + (DPTR))

MOVC A, @ A+ PC

(PC) ← (PC) + 1

(A) ← ((A) + (PC))

3.Address-object transfers

MOV DPTR, # DATA 16

The data pointer is loaded with the 16-bit constant indicated. No flags are affected.

(DPTR) ← #data 16

4.5.2 Arithmetic Instructions

The 8051 provides the basic mathematical operations like addition, subtraction, multiplication, division, increment, decrement, etc. The overflow flag permits the addition and

subtraction operations to serve both unsigned and signed binary integers. A correction operation is also provided to allow the arithmetic to be performed directly on the packed decimal (BCD) representations.

Some instructions like INC (increment), DEC (decrement) may also be used to modify the output port. In such cases the value used as the original port data will be read from the output data latch, not from the input pins.

1.Addition

There are four addition operations.

- ADD performs an addition between the register A and the second source operand.
- ADDC (add with carry) performs an addition between the register A and the second source operand; adds 1 if the C flag is found previously set.
- DA (decimal-add-adjust for BCD addition) performs a correction to the sum which resulted from the binary addition of two two-digit decimal operands. The packed decimal sum formed by DA is returned to A. The carry flag is set if the BCD result is greater than 99; or else it is cleared.
- INC (increment) performs an addition of the source operand and 1.

ADD A, <src.byte>

ADD adds the byte variables indicated to the accumulator, leaving the result in the accumulator. Four source operand addressing modes are allowed: register, direct, register indirect, and immediate.

```
ADD   A, Rn
        (A) <— (A) + (Rn)
ADD   A, direct
        (A) ← (A) + direct
ADD   A, @Ri                    (where Ri = R0 or R1)
        (A) <— (A) + ((Ri))
ADD   A, #data
        (A) ← (A) + #data
```

ADDC A, <src.byte> (Add with carry)

The ADDC simultaneously adds the byte variables indicated, the carry flag and the accumulator contents, leaving the result in the accumulator.

```
ADDC   A, Rn
        (A) ← (A) + (C) + (Rn)
ADDC   A, direct
        (A) ← (A) + (C) + (direct)
ADDC   A, @Ri  (where Ri = R0 or R1)
        (A)  ← (A) + (C) + ((Ri))
```

ADDC A, #data

(A) <— (A) + (C) + #data

DA A (Decimal-adjust accumulator for addition)

The DA A adjusts the 8-bit value in the accumulator, resulting from the earlier addition of two variables (each in packed-BCD format), producing two 4-bit digits. Any ADD or ADDC instruction may have been used to perform the addition. The algorithm followed is as follows:

1. If the value of the lower nibble in ACC is greater than 9, or if AC flag is set, then 6 is added to ACC.

2. If the value of the higher nibble is now greater than 9, or if CY flag is set, then 6 is added to the higher nibble of ACC.

All flags are affected.

If [[A3-0 > 9] V [[AC0 =1]]

Then (A3-0) ← (A3-0) + 6

and

If [[A7-4 > 9] V [(C) =1]]

Then (A7-4) ← (A7-4) + 6

INC <byte> (Increment)

INC increments the indicated variable by 1. No flags are affected.

INC A

(A) ← (A) + 1

INC Rn

(Rn) ← (Rn) + 1

INC direct

(direct) ← (direct) + 1

INC @ Ri

((Ri)) ← ((Ri)) + 1

INC DPTR (Increment data pointer)

Increments the 16-bit data pointer by 1. No flags are affected.

(DPTR) ← (DPTR) + 1

2.Subtraction

There are two subtraction operations.

- SUBB (subtract with borrow) performs a subtraction of the second source operand from the first operand (the accumulator), subtracts 1 if the C flag is found previously set.
- DEC (decrement) performs a subtraction of 1 from the source operand.

SUBB A, <src-byte> (Subtract with borrow)

SUBB subtracts the indicated variable and the carry flag together from the accumulator, leaving the result in the accumulator.

SUBB A, Rn

(A) ← (A) – (C) – (Rn)

SUBB A, direct

(A) ← (A) – (C) – (direct)

SUBB A, @ Ri

(*A*) ← (A) – (C) – (Ri)

SUBB A, #data

(B) ← (A) – (C) – #data

DEC byte (Decrement)

The variable indicated is decremented by 1. No flags are affected.

DEC A

(A) ← (A) - 1

DEC Rn

(Rn)← (Rn) - 1

DEC direct

(direct) ← (direct) - 1

DEC @ Ri

((Ri) ← (Ri) -1

3.Multiplication

MUL performs an unsigned multiplication of register A by register B, returning a double-byte result.

MUL AB

The MUL AB multiplies the unsigned 8-bit integers in the accumulator and register B. The low-byte of the 16-bit product is left in the accumulator, and the high-order byte in register B. If the product is greater than 255 (OFFH), the overflow flag is set; otherwise it is cleared. The carry flag is always cleared.

(A) 7-0 ← (A) x (B)

(B) 15 -8

4.Division

The DIV performs an unsigned division of register A by register B.

DIV AB

The DIV AB divides the unsigned 8-bit integer in the accumulator by the unsigned 8-bit integer in register B. The accumulator receives the integer part of the quotient; register B receives the integer remainder. The carry and the overflow flags will be cleared. An exception is: if B had originally contained 00H, the values returned to the accumulator and B register will be undefined and the overflow flag will be set. The carry flag is cleared in any case.

(A) 15 – 8

:

: ← (A)/(B)

:

(B) 7- 0

4.5.3 Logic instructions

8051 performs the basic logic operations on both bit and byte operands.

1.Single operand instructions:

There are seven single-operand logical operations as follows:

- CLR is used to set either the register A the carry, or any direct addressed bit to O.
- SETB sets either the carry or any direct addressed bit to 1.
- CPL either forms the 1's complement of the operand in register A or the 1's complement of the carry or any direct addressed bit.
- RL, RLC, RR, RRC, SWAP. Five rotate operations can be performed on register A. RL (rotate left), RR (rotate right), RLC (rotate left through carry), RRC (rotate right through carry), and SWAP (swap nibbles). For RLC and RRC the C flag becomes equal to the last bit rotated out. SWAP rotates the register A four places left to exchange bits 3 through 0 with bits 7 through 4.

The CPL bit and CLR bit instructions may be used to modify an output pin. In such cases, the value used as the original data will be read from the output data latch, not the input pin.

CLR A (Clear accumulator)

The accumulator is cleared (all bits are set to 0). No flags are affected.

(A) ← 0

CLR bit (clear bit)

The indicated bit is cleared (reset to 0). No other flags are affected. CLR can operate on the carry flag or any directly addressable bit.

CLR C

$(C) \leftarrow 0$

CLR bit

(bit) ← 0

SETB <bit> (set bit)

The SETB sets the indicated bit to 1. The SETB can operate on the carry flag or any directly addressable bit. No other flags are affected.

SETB C

(B) ← 1

SETB bit

(bit) ← 1

CPL A (Complement accumulator)

Each bit of the accumulator is logically complemented (1's complement). No flags are affected.

(A) ← $\overline{A}$

CPL bit (Complement bit)

The bit variable specified is complemented. A bit which had been 1 is changed to 0 and vice-versa. No other flags are affected. The CPL can operate on the carry or any directly addressable bit.

CPL C

(C) ← ($\overline{C}$)

CPL bit

(bit) ← ($\overline{bit}$)

RL A (Rotate accumulator left)

The eight bits in the accumulator are rotated 1 bit to the left. Bit7 is rotated into the bit 0 position. No flags are affected.

(An+1) ← (An) n=0 – 6

(A0) ← (A7)

RLC A (Rotate accumulator left through the carry flag)

The eight bits in the accumulator and the carry flag are together rotated 1 bit to the left. Bit 7 moves into the carry flag; the original state of the carry flag moves into the bit 0 position. No other flags are affected.

(An+1) ← (An) n=0 – 6

(A0) ← (C)

(C) ← (A7)

RR A (Rotate accumulator right)

The eight bits in the accumulator are rotated 1 bit to the right. Bit7 is rotated into the bit 0 position. No flags are affected.

(An) ← (An + 1) n=0 – 6

(A7) ← (A0)

RRC A (Rotate accumulator right through the carry flag)

The eight bits in the accumulator and the carry flag are together rotated 1 bit to the right. Bit 0 moves into the carry flag; the original state of the carry flag moves into the bit 7 position. No other flags are affected.

(An) ← (An + 1) n=0 – 6

(A7) ← (C)

(C) ← (A0)

SWAP A (Swap nibbles within the accumulator)

SWAP A interchanges the low and high-order nibbles (4-bit fields) of the accumulator (bits 3-0 and bits 7-4). The operation can also be thought of as a 4-bit rotate instruction. No flags are affected.

(A3-0) ↔ (A7-4)

1.Two operand instructions:

Three two-operand logical instructions ANL, ORL and XRL are provided to perform AND, OR and XOR operations respectively on bit as well as byte operands.

These instructions may be used to modify an output port. In such cases, the value used as the original port data will be read from the output data latch, not the input pins.

ANL <destbyte>, <src.byte> (Logical AND for byte variables)

ANL performs the bit-wise logical AND operation between the variables indicated and stores the results in the destination variable. No flags are affected.

The two operands allow six addressing mode combinations. When the destination is the accumulator, the source can use register, direct, register indirect, or immediate addressing; when the destination is a direct address, the source can be the accumulator or the immediate data.

ANL A, Rn

(A) ← (A) $\wedge$ (Rn)

ANL A, direct

(A) ← (A) $\wedge$ (direct)

ANL A, @ Ri

(A) ← (A) $\wedge$ ((Ri))

ANL A, # data

(A) ← (A) $\wedge$ #data

ANL C, <src.bit> (Logical AND for bit variables)

The carry flag is modified by ANDing with the source bit or its logical complement. No other flags are affected. Only the direct bit addressing is allowed for the source operand.

ANL C, bit

(C) ← (C) □ (bit)

ORL <dest-byte>, <src-byte> (Logical OR for byte variables)

The ORL performs the bit-wise logical OR operation between the indicated variables, storing the results in the destination byte. No flags are affected. The two operands allow six addressing mode combinations. When the destination is the accumulator, the source can use register, direct, register indirect, or immediate addressing; when the destination is a direct address, the source can be the accumulator or immediate data.

ORL A, Rn

(A) ← (A) V (Rn)

ORL A, direct

(A) ← (A) V (direct)

ORL A, @ Ri

(A) ← (A) V ((Ri))

ORL A, # data

(A) ← (A) V #data

ORL C, <src-bit> (Logical OR for bit Variables)

The carry flag is modified by ORing with the source bit or its logical complement. No other f lags are affected.

ORL C, /bit

$(C) \leftarrow (C)\ V\ (\overline{bit})$

XRL <dest-byte>, <src-byte> (Logical XOR for byte variables)

The XRL performs the bit-wise logical XOR operation between the indicated variables, storing the results in the destination. No flags are affected. The two operands allow six addressing mode combinations. When the destination is the accumulator, the source can use

register, direct, register indirect, or immediate addressing. When the destination is a direct address, the source can be the accumulator or immediate data.

XRL A, Rn

(B)← (A) ∀ (Rn)

XRL A, direct

(B) ← (A) ∀ (direct)

XRL A, @ Ri

(B) ← (A) ∀ ((Ri))

XRL A, # data

(B) ← (A) ∀ #data

(C)

4.5.4 Control Transfer instructions

There are three classes of control transfer operations.

1. Unconditional calls, returns and jumps
2. Conditional jumps
3. Interrupts

1.Unconditional calls, returns and jumps

ACALL addr 11 (absolute call)

ACALL unconditionally calls a subroutine located at the indicated address. Since ACALL is a 2-byte instruction, PC is incremented by 2 to point to the next instruction. The destination address is obtained by successively concatenating the five high-order bits of the incremented PC, op-code bits 7-5, and the second byte of the instruction. The subroutine called must, therefore, start within the same 2K block of the program memory as the first byte of the instruction following ACALL. No flags are affected

(PC) ← (PC) + 2
(SP) ← (SP) + 1
((SP) ← (PC 7-0)
(SP) ← (SP) + 1
((SP) ← (PC 15-8)
(PC10-0) ← page address

LCALL addrl6 (Long call)

The LCALL calls a subroutine located at the indicated address. Since LCALL is a 3-byte instruction, PC is incremented by 3 to point to the next instruction. The destination address is mentioned in absolute term in the instruction as addr16. No flags are affected.

(PC) ← (PC) + 3
(SP) ← (SP) + 1
((SP) ← (PC 7-0)
(SP) ← (SP) + 1
((SP) ← (PC 15-8)
(PC10-0) ← page address

RET (Return from subroutine)

The RET pops the return address from the stack and loads into the PC. Program execution continues at the resulting address. No flags are affected.

(PC 15-8) ← ((SP))
(SP) ← (SP) – 1
(PC 7-0) ← ((SP))
(SP) ← (SP) – 1

RETI (Return from interrupt)

The RETI pops the return address from the stack and loads into the PC and restores the interrupt logic to accept additional interrupts at the same priority level as the one just processed. Program execution continues at the resulting address.

(PC 15-8) ← ((SP))
(SP) ← (SP) – 1
(PC 7-0) ← ((SP))
(SP) ← (SP) – 1

AJMP addrll (Absolute jump)

The AJMP transfers the program execution to the indicated address, which is formed at runtime by concatenating the high-order five bits of the PC (after incrementing the PC twice), opcode bits 7-5, and the second byte of the instruction. The destination must, therefore, be within the same 2K block of the program memory as the first byte of the instruction following AJMP.

(PC) ← (PC) + 2
(PC10-0) ← page address

LJMP addrl6 (Long jump)

The LJMP causes an unconditional branch to the indicated address. No flags are affected.

(PC) ← addr 15-0

SJMP rel (Short jump)

Program control branches unconditionally to the address indicated. The branch destination is computed by adding the signed displacement in the second instruction byte to the PC, after incrementing the PC twice. Therefore, the range of destinations allowed is from 128 bytes preceding this instruction to 127 bytes following it.

(PC) ← (PC) + 2
(PC) ← (PC) + rel

JMP @ A+ DPTR (Jump indirect)

The 8-bit unsigned contents of the accumulator are added to the 16- bit data pointer, and the resulting sum is loaded to the program counter. No flags are affected.

(PC) ← (A) + (DPTR)

NOP (No operation)

Execution continues at the following instructions. Other than PC, no registers or flags are affected.

(PC) ← (PC) + 1 1

2.Conditional Jumps:

JB bit, rel (Jump if bit set)

If the indicated bit is a 1, jump to the address indicated; otherwise, proceed with the next instruction. The bit tested is not modified. No flags are affected.

(PC) ← (PC) + 3
If (bit) =1
Then
(PC) ← (PC) + rel

JBC bit, rel (Jump if bit is set and clear bit)

If the indicated bit is a 1, branch to the address indicated; otherwise, proceed with the next instruction. In either case, clear the designated bit. No flags are affected.

When this instruction is used to test an output pin, the value used as the original data will be read from the output data latch, not from the input pin.

(PC) ← (PC) + 3
If (bit) =1
Then
(bit) ← 0
(PC) ← (PC) + rel

JC rel (Jump if carry is set)

If the carry flag is set, branch to the address indicated; otherwise proceed with the next instruction. No flags are affected

(PC) ← (PC) + 2
If (C) =1
Then
(PC) ← (PC) + rel

JNB bit, rel (Jump if bit not set)

If the indicated bit is a 0, branch to the indicated address; otherwise proceed with the next instruction. The bit tested is not modified. No flags are affected.

(PC) ← (PC) + 3
If (bit) = 0
Then
(PC) ← (PC) + rel

JNC rel (Jump if carry not set)

If the carry flag is a 0, branch to the address indicated; otherwise, proceed with the next instruction. The carry flag is not modified.

(PC) ← (PC) + 2
If (C) =0
Then
(PC) ← (PC) + rel

JNZ rel (Jump if accumulator not zero)

If any bit of the accumulator is a 1, branch to the indicated address; otherwise, proceed with the next instruction. The accumulator is not modified. No flags are affected.

(PC) ← (PC) + 2
If (A) ≠ 0
Then
(PC) ← (PC) + rel

JZ rel (Jump if accumulator zero)

If all the bits of the accumulator are 0, branch to the address indicated; otherwise, proceed with the next instruction. The accumulator is not modified. No flags are affected.

(PC) ← (PC) + 2
If (A) = 0
Then
(PC) ← (PC) + rel

CJNE < dest byte>, <src byte> (Compare and jump if not equal)

The CJNE compares the magnitudes of the first two operands, and branches if their values are not equal. The carry flag is set, if the unsigned integer value of <dest-byte> is less than the signed integer value of <src-byte>; otherwise, the carry is cleared. Neither operand is affected. The first two operands allow four addressing mode combinations—the accumulator ay be compared with any directly addressed byte or immediate data and any indirect RAM location, or the working register can be compared with an immediate data.

CJNE A, direct, rel

(PC) ← (PC) + 3
If (direct) < (A)
then (PC) ← (PC) + rel and (C) ← 0
or
If (direct) > (A)
then (PC) ← (PC) + rel and (C) ← 1

CJNE A, #data, rel

(PC) ← (PC) + 3
If #data < (A)
then (PC) ← (PC) + rel and (C) ← 0
or
If #data > (A)
then (PC) ← (PC) + rel and (C) ← 1

DJNZ < byte>, <rel-addr> (Decrement and jump if not zero

The DJNZ decrements by 1 the contents of the location indicated, and branches to the address indicated by the second operand if the resulting value is not a zero. No flags are affected. The location (whose contents are decremented) may be a register or a directly addressed byte.

DJNZ Rn,rel

(PC) ← (PC) + 2
(Rn) ← (Rn) -1
If (Rn) > 0 or (Rn) < (0)
then
(PC) ← (PC) + rel

DJNZ direct,rel

(PC) ← (PC) + 2
(direct) ← (direct) -1
If (direct) > 0 or (direct) < (0)
then
(PC) ← (PC) + rel

4.5.5 Single-Bit instructions

Instruction	Function
SETB bit	Set the bit (bit = 1)
CLRbit	Clear the bit (bit = 0)
CPL bit	Complement the bit (bit = NOT bit)
JBbit,target	Jump to target if bit=1 (jump if bit)
JNB bit,target	Jump to target if bit= 0(jump if no bit)
JBCbit,target	Jump to target if bit = 1, clear bit (jump if bit, then clear)

4.6 ADDRESSING MODES

The CPU can access data in various ways. The data could be in a register, or in memory, or be provided as an immediate value. These various ways of accessing data are called addressing modes.

The various addressing modes of a microprocessor are determined when it is designed, and therefore cannot be changed by the programmer. The 8051 provides a total of five distinct addressing modes. They are as follows:

1. Immediate
2. Register
3. Direct .
4. Register Indirect
5. Indexed

4.6.1 Immediate and Register Addressing Modes

a) Immediate addressing mode

In this addressing mode, the source operand is a constant. In immediate addressing mode, as the name implies, when the instruction is assembled, the operand comes immediately after the opcode. The immediate data must be preceded by the pound sign, "#". This addressing mode can be used to load information into any of the registers, including the DPTR register. Examples follow.

```
MOV     A, #25H          ;load 25H into A
```

```
MOV    R4,#62          ;load the decimal value 62 into R4
MOV    B,#40H          ;load 40H into B
MOV    DPTR,#4521H     ;DPTR=4512H
```

Although the DPTR register is 16-bit, it can also be accessed as two 8-bit registers, DPH and DPL, where DPH is the high byte and DPL is the low byte. Look at the following code.

```
MOV    DPTR,#2550H is the same as.
MOV    DPL,#50H
MOV    DPH, #25H
```

The following would produce an error since the value is larger than 16bits.

```
MOV    DPTR,#68975          ; illegal value > 65535 (FFFFFH)
```

b) Register addressing mode

Register addressing mode involves the use of registers to hold the data to be manipulated. Examples of register addressing mode are as follows

```
MOV    A,R0     ; copy the contents of R0 into A
MOV    R2,A     ; copy the contents of A into R2
ADD    A,R5     ; add the contents of R5 to contents of A
ADD    A,R7     ; add the contents of R7 to contents of A
MOV    R6,A     ;save accumulator in R6
```

It should be noted that the source and destination registers must match in size. In other words, coding "MOV DPTR, A" will give an error, since the source is an 8-bit register and the destination is a 16-bit register. See the following.

```
MOV  DPTR,#25F5H
MOV R7,DPL
MOV R6,DPH
```

We can move data between the accumulator and Rn (for n = 0 to 7) but movement of data between Rn and registers is not allowed. For example, the instruction "MOV R4 ,R7"is invalid. In the first two addressing modes, the operands are either inside one of the registers or tagged along with the instruction itself. The data to be processed is often in some memory location of RAM or in the code space of ROM. There are many ways to access this data.

4.6.2 Accessing Memory using various Addressing Modes

We can use direct or register indirect addressing modes to access data stored either in RAM or registers of the 8051.

a) Direct addressing

There are 128 bytes of RAM in the 8051. The RAM has been assigned addresses 00 to *7FH.* The following is a summary of the allocation of these 128bytes.

1. RAM locations 00 - lFH are assigned to the register banks and stack.
2. RAM locations 20 - 2FH are set aside as bit-addressable space to save single-bit data.
3. RAM locations 30 - 7FH are available as a place to save byte-sized data.

Although the entire 128 bytes of RAM can be accessed using direct addressing mode, it is most often used to access RAM locations 30 - 7FH. This is due to the fact that register bank locations are accessed by the register names of R0- R7, but there is no such name for other RAM locations. In the direct addressing mode, the data is in a RAM memory location whose

address is known, and this address is given as a part of the instruction. The "#" sign distinguishes between the two modes.

```
MOV   R0,40H  ;save content of RAM location 40H in R0
MOV   56H,A           ;save content of A in RAM location 56H
MOV   R4,7FH  ;move contents of RAM location 7FH to R4
```

Stack and direct addressing mode

Another major use of direct addressing mode is the stack. In the 8051 family, only direct addressing mode is allowed for pushing onto the stack. Therefore, an instruction such as "PUSH A" is invalid. Pushing the accumulator onto the stack must be coded as "PUSH 0E0H" where 0E0H is the address of register A. Similarly, pushing R3 of bank 0 is coded as "PUSH 03 ", Direct addressing mode must be used for the *POP* instruction as well. For example, "POP 04" will pop the top of the stack into R4 of bank 0.

b) Register indirect addressing mode

In the register indirect addressing mode, a register is used as a pointer to the data. If the data is inside the CPU, only registers R0 and Rl are used for this purpose. In other words, R2 - R7 cannot be used to hold the address of an operand located in RAM when using this addressing mode. When R0 and Rl are used as pointers, when they hold the addresses of RAM locations, they must be preceded by the "@" sign, as shown below.

```
MOV   A, @R0 ; move contents of RAM whose address is held by r0 into A
MOV   @R1, B ; move contents of B into RAM location whose address is held by R1
```

Advantage of register indirect addressing mode

One of the advantages of register indirect addressing mode is that it makes accessing data dynamic rather than static as in the case of direct addressing mode.

Limitation of register indirect addressing mode in the 8051

R0 and Rl are the only registers that can be used for pointers in register indirect addressing mode. Since R0 and Rl are 8 bits wide, their use is limited to accessing any information in the internal RAM (scratch pad memory of 30H - 7FH, or SFR).However, there are times when we need to access data stored in external RAM or in the code space of on-chip ROM. Whether accessing externally connected RAM or on-chip ROM, we need a 16-bit pointer. In such cases DPTR register is used.

(c) Indexed addressing mode and on-chip ROM access

Indexed addressing mode is widely used in accessing data elements of look-up table entries located in the program ROM space of the 8051. The instruction used for this purpose is

```
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. Because the data elements are stored in the program (code) space ROM of the 8051, the instruction MOVC is used instead of MOV. The "C" means code. In this instruction the contents of A are added to the 16-bit register DPTR to form the 16-bit address of the needed data.

Look-up table and the MOVC instruction

The look-up table is a widely used concept in microprocessor programming. It allows access to elements of a frequently used table with minimum operations. As an example, assume

that for a certain application we need x^2value in the range of 0 to 9. We can use a look-up table instead of calculating it.

In addition to being used to access program ROM,DPTR can be used to access memory externally connected to the 8051. Another register used in indexed addressing mode is the program counter. In many of the examples above, the MOV instruction was used for the sake of clarity, even though one can use an; instruction as long as that instruction supports the addressing mode. For example, the instruction "ADD A, @R0" would add the contents of the memory location pointed to by R0 to the contents of register A.

Indexed addressing mode and MOVX instruction

8051 has 64Kbytes of code space under the direct control of the Program Counter register: We just showed how to use the MOVC instruction to access a portion of this 64K-byte-codespace as data memory space. In many applications the size of program code does not leave any room to share the 64Kbyte code space with data. For this reason the 8051 has another 64K bytes of memory space set aside exclusively for data storage. This data memory space is referred to as *external memory* and it is accessed only by the MOVX instruction. In other words, the 8051has a total *of 128K* bytes of memory space since 64Kbytes of code added to 64K bytes of data space gives us 128Kbytes. One major difference between the code space and data space is that, unlike code space, the data space cannot be shared between code and data.

LIST OF QUESTIONS
PART A

1. List the features of 8051 microcontroller.

The resources offered by 8051 microcontroller are:

1. 8-bit CPU
2. On-chip oscillator
3. 4KB of ROM (Program memory)
4. 128 bytes of RAM (Data memory)
5. 21 Special function registers
6. 32 I/O lines (Ports P0 to P3)
7. 64 KB address space for external data memory
8. 64 KB address space for program memory
9. Two 16-bit timer/counter
10. Five source interrupt structure
11. Full duplex serial port
12. Bit addressability
13. Powerful bit processing capability

2. List the addressing modes of 8051.

The various addressing modes of 8051 are as follows:

1. Immediate
2. Register