

## UNIT-7

### 8051 MICROCONTROLLER AND ITS PROGRAMMING

#### **INTRODUCTION**

The microcontroller incorporates all the features that are found in microprocessor. The microcontroller has built in ROM, RAM, Input Output ports, Serial Port, timers, interrupts and clock circuit. A microcontroller is an entire computer manufactured on a single chip. Microcontrollers are usually dedicated devices embedded within an application. For example, microcontrollers are used as engine controllers in automobiles and as exposure and focus controllers in cameras. In order to serve these applications, they have a high concentration of on-chip facilities such as serial ports, parallel input output ports, timers, counters, interrupt control, analog-to-digital converters, random access memory, read only memory, etc. The I/O, memory, and on-chip peripherals of a microcontroller are selected depending on the specifics of the target application. Since microcontrollers are powerful digital processors, the degree of control and programmability they provide significantly enhances the effectiveness of the application. The 8051 is the first microcontroller of the MCS-51 family introduced by Intel Corporation at the end of the 1970s. The 8051 family with its many enhanced members enjoys the largest market share, estimated to be about 40%, among the various microcontroller architectures.

The microcontroller has on chip peripheral devices. In this unit firstly we differentiate microcontroller from microprocessor then we will discuss about Hardware details of 8051 and then introduce the Assembly level language in brief.

#### **Microcontrollers**

Microcontroller (MC) may be called computer on chip since it has basic features of microprocessor with internal ROM, RAM, Parallel and serial ports within single chip. Or we can say microprocessor with memory and ports is called as microcontroller. This is widely used in washing machines, VCD player, microwave oven, robotics or in industries.

#### **Why the name Micro controller?**

Basically used for control actions. It is used to control the operation of machine using fixed program that is stored in ROM/EPROM and that does not change over the life time.

#### **Microcontrollers Vs Microprocessors**

- A microprocessor requires an external memory for program/data storage. Instruction execution requires movement of data from the external memory to the microprocessor or vice versa. Usually, microprocessors have good computing power and they have higher clock speed to facilitate faster computation.
- A microcontroller has required on-chip memory with associated peripherals. A microcontroller can be thought of a microprocessor with inbuilt peripherals.
- A microcontroller does not require much additional interfacing ICs for operation and it functions as a standalone system. The operation of a microcontroller is multipurpose, just like a Swiss knife.
- Microcontrollers are also called embedded controllers. A microcontroller clock speed is limited only to a few tens of MHz. Microcontrollers are numerous and many of them are application specific.

#### **Development/Classification of microcontrollers**

Microcontrollers have gone through a silent evolution (invisible). The evolution can be rightly termed as silent as the impact or application of a microcontroller is not well known to a common user, although microcontroller technology has undergone significant change since early 1970's. Development of some popular microcontrollers is given as follows.

Intel 4004	4 bit (2300 PMOS trans, 108 kHz)	1971
Intel 8048	8 bit	1976

Intel 8031	8 bit (ROM-less)	.
Intel 8051	8 bit (Mask ROM)	1980
Microchip PIC16C64	8 bit	1985
Motorola 68HC11	8 bit (on chip ADC)	.
Intel 80C196	16 bit	1982
Atmel AT89C51	8 bit (Flash memory)	.
Microchip PIC 16F877	8 bit (Flash memory + ADC)	.

We use more number of microcontrollers compared to microprocessors. Microprocessors are primarily used for computational purpose, whereas microcontrollers find wide application in devices needing real time processing/control. Applications of microcontrollers are numerous. Starting from domestic applications such as in washing machines, TVs, air conditioners, microcontrollers are used in automobiles, process control industries, cell phones, electrical drives, and robotics and in space applications.

### Features of Modern Microcontrollers

- Built-in Monitor Program
- Built-in Program Memory
- Interrupts
- Analog I/O
- Serial I/O
- Facility to Interface External Memory
- Timers

### DIFFERENCE BETWEEN MICROCONTROLLER AND MICROPROCESSOR

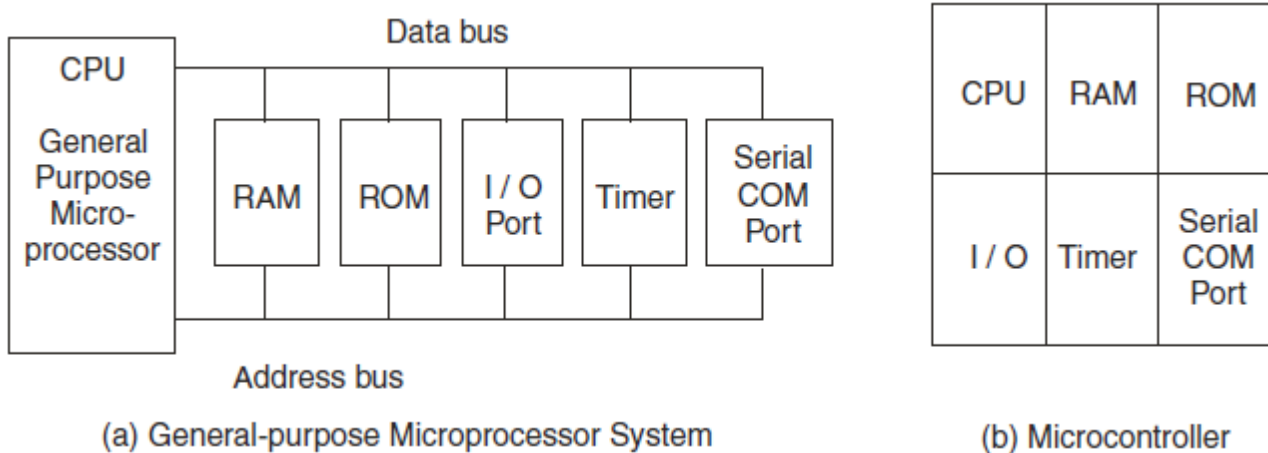


Fig. Structure of Microprocessor and Microcontroller

It is very clear from figure that in microprocessor we have to interface additional circuitry for providing the function of memory and ports, for example we have to interface external RAM for data storage, ROM for program storage, programmable peripheral interface (PPI) 8255 for the Input Output ports, 8253 for timers, USART for serial port. While in the microcontroller RAM, ROM, I/O ports, timers and serial communication ports are in built. Because of this it is called as "system on chip". So in micro-controller there is no necessity of additional circuitry which is interfaced in the microprocessor because memory and input output ports are inbuilt in the microcontroller. Microcontroller gives the satisfactory performance for small applications. But for large applications the memory requirement is limited because only 64 KB

memory is available for program storage. So for large applications we prefer microprocessor than microcontroller due to its high processing speed.

### Advantages of microcontroller:

- Low cost
- Small size of product
- Easy to troubleshoot and maintain
- More reliable
- Additional memory, I/o can also be added
- Software security feature
- All features available with 40 pins.
- Useful for small dedicated applications and not for larger system designs which may require many more I/O ports.
- Mostly used to implement small control functions.

### ✚ CRITERIA FOR SELECTION OF A MICROCONTROLLER IN EMBEDDED SYSTEM

Criteria for selection of microcontroller in any embedded system is as following:

(a) Meeting the computing needs of task at hand efficiently and cost effectively

- Speed of operation
- Packing
- Power consumption
- Amount of RAM and ROM on chip
- No. of I/O pins and timers on chip
- Cost

(b) Availability of software development tools such as compiler, assembler and debugger.

### ✚ INTERNAL STRUCTURE OF A MICROCONTROLLER

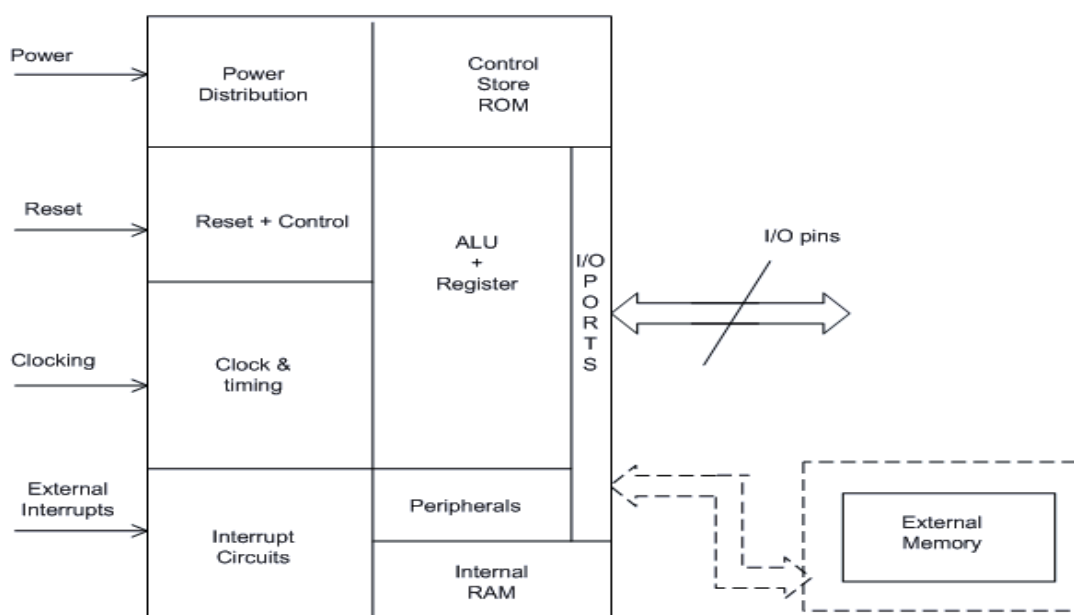


Fig. Internal Structure of a Microcontroller

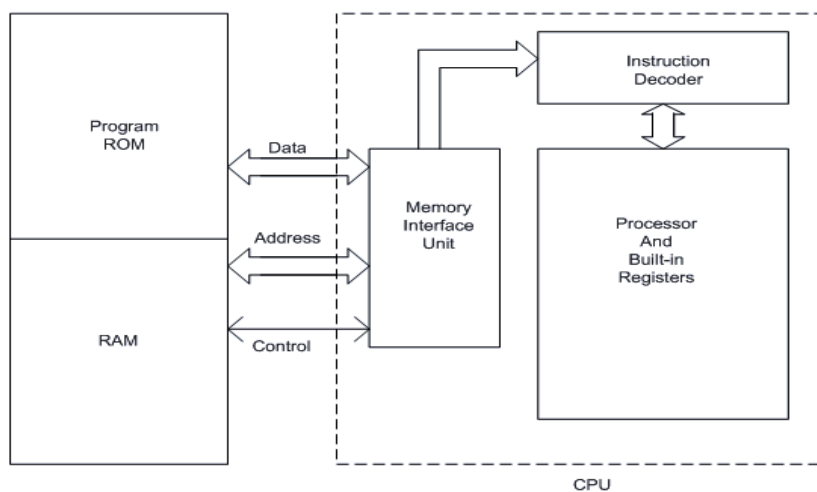
At times, a microcontroller can have external memory also (if there is no internal memory or extra memory interface is required). Early microcontrollers were manufactured using bipolar or NMOS technologies. Most modern microcontrollers are manufactured with CMOS technology, which leads to reduction in size and power loss. Current drawn by the IC is also reduced considerably from 10mA to a few micro Amperes in sleep mode (for a microcontroller running typically at a clock speed of 20MHz).

**Harvard vs. Princeton Architecture**

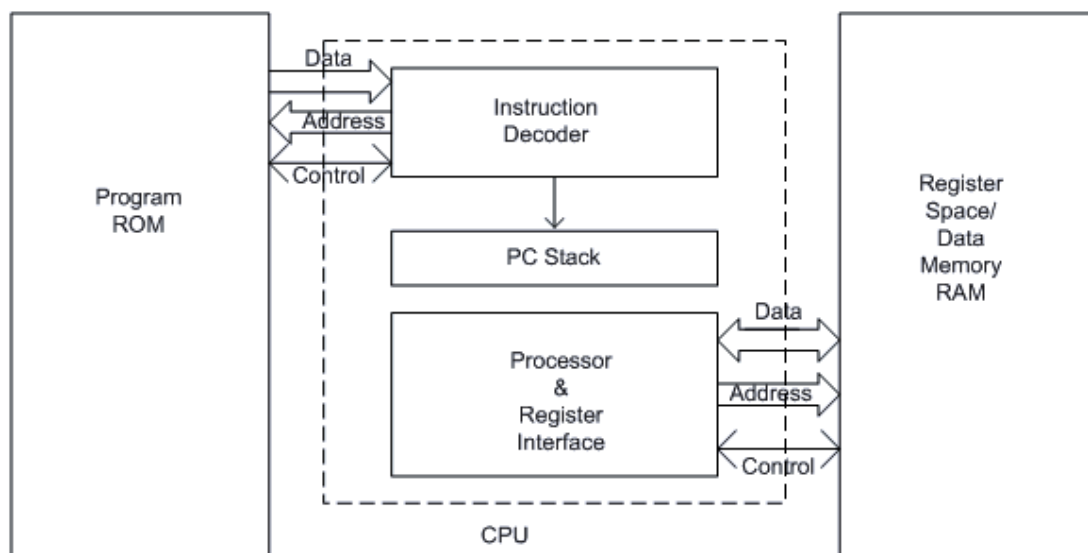
Many years ago, in the late 1940's, the US Government asked Harvard and Princeton universities to come up with a computer architecture to be used in computing distances of Naval artillery shell for defense applications. Princeton suggested computer architecture with a single memory interface. It is also known as Von Neumann architecture after the name of the chief scientist of the project in Princeton University John Von Neumann (1903 - 1957 Born in Budapest, Hungary).

Harvard suggested a computer with two different memory interfaces, one for the data / variables and the other for program / instructions. Although Princeton architecture was accepted for simplicity and ease of implementation, Harvard architecture became popular later, due to the parallelism of instruction execution.

**Princeton Architecture (Single memory interface)**



**Harvard Architecture (Separate Program and Data Memory interfaces)**



Some of the microcontrollers of 8051 family are given as follows:

DEVICE	ON-CHIP DATA MEMORY (bytes)	ON-CHIP PROGRAM MEMORY (bytes)	16-BIT TIMER/COUNTER	NO. OF VECTORED INTERRUPTS	FULL DUPLEX I/O
8031	128	None	2	5	1
8032	256	none	2	6	1
8051	128	4k ROM	2	5	1
8052	256	8k ROM	3	6	1
8751	128	4k EPROM	2	5	1
8752	256	8k EPROM	3	6	1
AT89C51	128	4k Flash Memory	2	5	1
AT89C52	256	8k Flash memory	3	6	1

## INTEL 8051 MICROCONTROLLER

### Introduction:

The **Intel MCS-51** (commonly referred to as **8051**) is a Harvard architecture, single chip microcontroller ( $\mu\text{C}$ ) series which was developed by Intel in 1980 for use in embedded systems. Intel's original versions were popular in the 1980s and early 1990s. Intel's original MCS-51 family was developed using NMOS technology, but later versions, identified by a letter C in their name (e.g., 80C51) used CMOS technology and consume less power than their NMOS predecessors. This made them more suitable for battery-powered devices. The 8051 architecture provides many functions (CPU, RAM, ROM, I/O, interrupt logic, timer, etc.) in a single package.

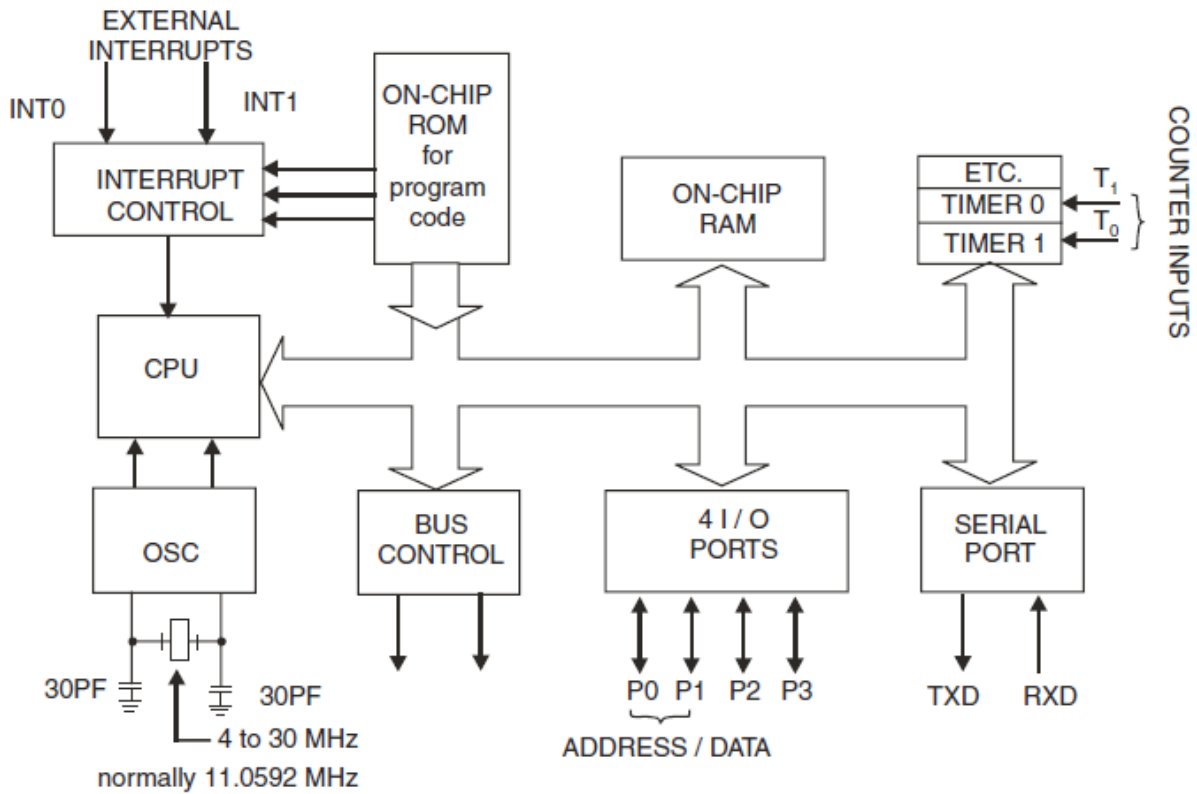
The basic architecture of 8051 is given in fig below

### Features:

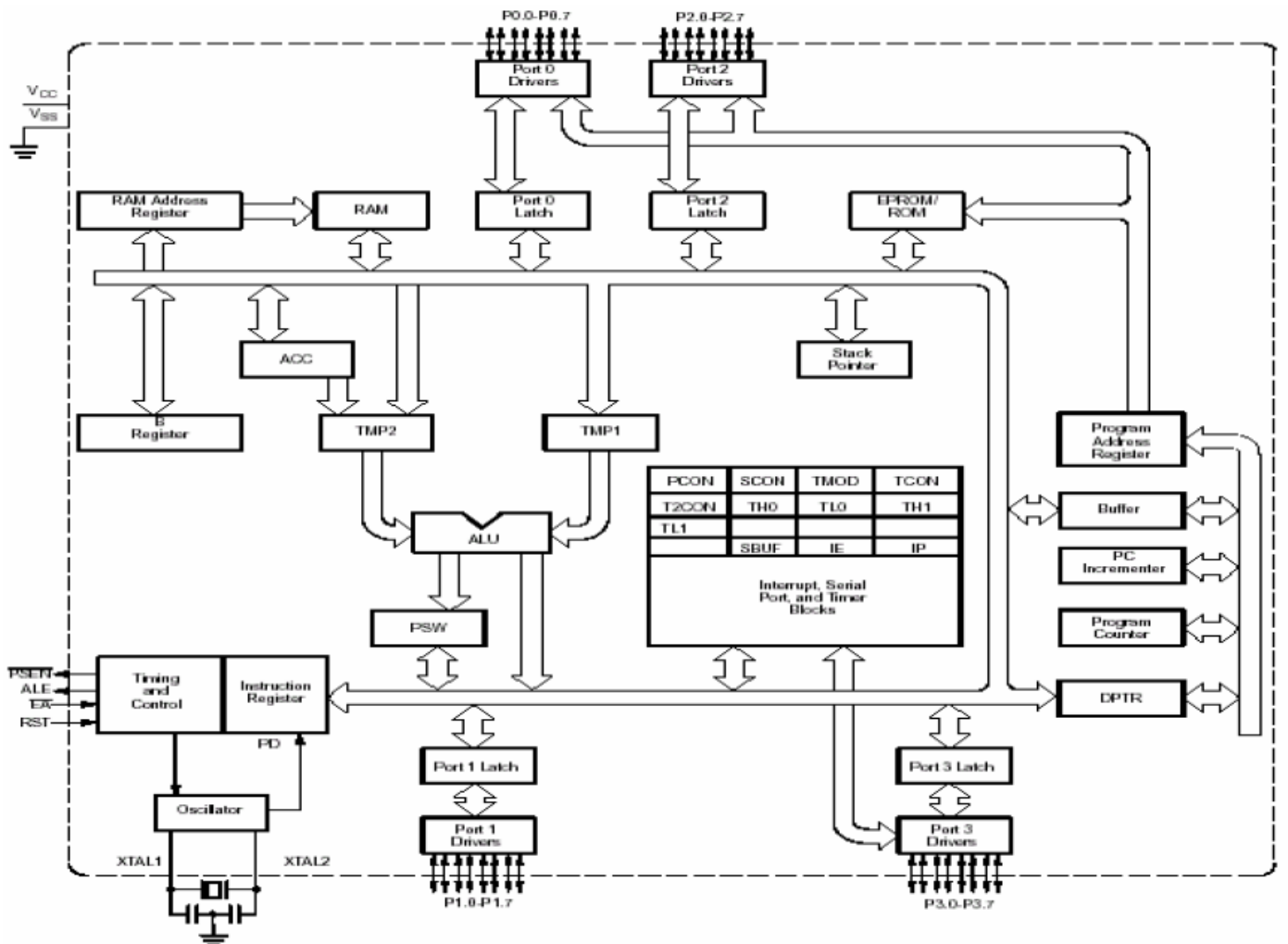
Various features of 8051 microcontroller are given as follows.

- 8-bit ALU, Accumulator, 8-bit Registers and 8-bit data bus; hence it is an 8-bit microcontroller 16-bit Program Counter.
- 8-bit Processor Status Word (PSW)
- 8-bit Stack Pointer
- Internal RAM of 128bytes
- On chip ROM is 4KB
- Special Function Registers (SFRs) of 128 bytes
- 32 I/O pins arranged as four 8-bit ports (P0 - P3)
- Two 16-bit timer/counters : T0 and T1
- Two external and three internal vectored interrupts
- UART (serial port)

**Simple Block Diagram of 8051:**



**Internal Architecture of 8051 Microcontroller**



SI100529

**Block Diagram Description:****Accumulator (Acc):**

- Operand register
- Implicit or specified in the instruction
- Has an address in on chip SFR bank

**B Register:** Used to store one of the operands for multiplication and division, otherwise, scratch pad considered as a SFR.

**Program Status Word (PSW):** Set of flags contains status information.

**Stack Pointer (SP):** 8 bit wide register. Incremented before data is stored on to the stack using PUSH or CALL instructions. Stack defined anywhere on the 128 byte RAM

**Data Pointer (DPTR):** 16 bit register contains DPH and DPL Pointer to external RAM address. DPH and DPL allotted separate addresses in SFR bank

**Port 0 To 3 Latches & Drivers:** Each I/O port allotted a latch and a driver Latches allotted address in SFR. User can communicate via these ports P0, P1, P2, and P3.

**Serial Data Buffer:** Internally had TWO independent registers, TRANSMIT buffer (parallel in serial out – PISO) and RECEIVE buffer (serial in parallel out –SIPO) identified by SBUF and allotted an address in SFR.

**Timer Registers:** for Timer0 (16 bit register – TLO & TH0) and for Timer1 (16 bit register – TL1 & TH1) four addresses allotted in SFR

**Control Registers:** Control registers are IP, IE, TMOD, TCON, SCON, and PCON. These registers contain control and status information for interrupts, timers/counters and serial port. Allotted separate address in SFR.

**Timing and Control Unit:** This unit derives necessary timing and control signals for internal circuit and external system bus

**Oscillator:** generates basic timing clock signal using crystal oscillator.

**Instruction Register:** decodes the opcode and gives information to timing and control unit.

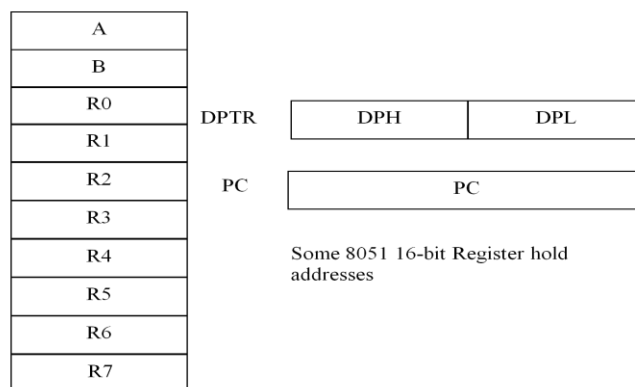
**EPROM & program address Register:** provide on chip EPROM and mechanism to address it. All versions don't have EPROM.

**Ram & Ram Address Register:** provide internal 128 bytes RAM and a mechanism to address internally

**ALU:** Performs 8 bit arithmetic and logical operations over the operands held by TEMP1 and TEMP 2. User cannot access temporary registers.

**SFR Register Bank:** set of special function registers address range: 80 H to FF H. Interrupt, serial port and timer units control and perform specific functions under the control of timing and control unit

### REGISTER SET OF 8051



Some 8-bit Registers of the 8051

Some 8051 16-bit Register hold addresses

**Accumulator**

ACC is the Accumulator register. The mnemonics for accumulator-specific instructions, however, refer to the accumulator simply as A.

**B Register**

The B register is used during multiply and divide operations. For other instructions it can be treated as another scratch pad register.

**Program Status Word**

The PSW register contains program status information as detailed in Table below

**Table .PSW: Program Status Word Register**

CY	AC	F0	RS1	RS0	OV	–	P
----	----	----	-----	-----	----	---	---

CY	PSW.7	Carry flag.
AC	PSW.6	Auxiliary carry flag.
F0	PSW.5	Available to the user for general purpose.
RS1	PSW.4	Register Bank selector bit 1.
RS0	PSW.3	Register Bank selector bit 0.
OV	PSW.2	Overflow flag.
–	PSW.1	User-definable bit.
P	PSW.0	Parity flag. Set/cleared by hardware each instruction cycle to indicate an odd/even number of 1 bits in the accumulator.

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

**Stack Pointer**

The Stack Pointer register is 8 bits wide. It is incremented before data is stored during PUSH and CALL executions. While the stack may reside anywhere in on-chip RAM, the Stack Pointer is initialized to 07H after a reset. This causes the stack to begin at location 08H.

**Data Pointer**

The Data Pointer (DPTR) consists of a high byte (DPH) and a low byte (DPL). Its intended function is to hold a 16-bit address. It may be manipulated as a 16-bit register or as two independent 8-bit registers.

**Ports 0 to 3**

P0, P1, P2 and P3 are the SFR latches of Ports 0, 1, 2 and 3, respectively.

**Serial Data Buffer**

The Serial Data Buffer is actually two separate registers, a transmit buffer and a receive buffer register. When data is moved to SBUF, it goes to the transmit buffer where it is held for serial transmission. (Moving a byte to SBUF is what initiates the transmission.) When data is moved from SBUF, it comes from the receive buffer.

**Timer Registers**

Register pairs (TH0, TL0), (TH1, TL1), and (TH2, TL2) are the 16-bit counting registers for Timer/Counter 0, 1, and 2, respectively.

**Capture Registers**

The register pair (RCAP2H, RCAP2L) are the capture register for the Timer 2 'capture mode'. In this



mode, in response to a transition at the 80C52's T2EX pin, TH2 and TL2 are copied into RCAP2H and RCAP2L. Timer 2 also has a 16-bit auto-reload mode, and RCAP2H and RCAP2L hold the reload value for this mode.

**Control Registers:** Special Function Registers IP, IE, TMOD, TCON, T2CON, SCON, and PCON contain control and status bits for the interrupt system, the timer/counters, and the serial port.

### Special Function Registers (SFR)

The set of Special Function Registers (SFRs) contains important registers such as Accumulator, Register B, I/O Port latch registers, Stack pointer, Data Pointer, Processor Status Word (PSW) and various control registers. Some of these registers are bit addressable. Addresses from 80H to FFH of all Special Function Registers

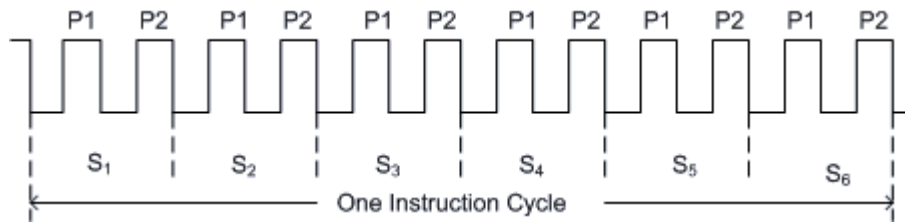
- PSW, P0-P3, IP, IE, TCON, SCON
  - Bit addressable, 8bit each, 11 in number
- SP, DPH, DPL, TMOD, TH0, TL0, TH1, TL1, SBUF, PCON
  - Byte addressable, 8bit each.
  - DPTR – data pointer, accesses external memory. DPH + DPL = DPTR
- Starting 32 bytes of RAM – general purpose registers, divided into 4 register banks of 8 registers each. Only one of these banks accessible at one time. RS1 and RS0 of PSW used to select bank.
- TH0-TL0 and TH1-TL1
  - 16 bit timer registers
- P0-P3 – port latches
- SP, PSW, IP – Interrupt Priority, IE – enable
- TCON – timer/counter control register to turn on/off the timers, interrupt control flags for external interrupts like INT<sub>1</sub> and INT<sub>0</sub>
- TMOD – modes of operation of timer/counter
- SCON – serial port mode control register
- SBUF – serial data buffer for transmit and receive
- PCON – Power control register – power down bit, idle bit

**Table 17.3** SFR Registers, their Addresses and Contents after Reset

Register	Bit Addressable	Address (SFR)	Content After Reset
ACC	Y	0E0H	0000 0000
B	Y	0F0H	0000 0000
PSW	Y	0D0H	0000 0000
SP	N	81H	0000 0111
DPH	N	82H	0000 0000
DPL	N	83H	0000 0000
P0	Y	80H	1111 1111
P1	Y	90H	1111 1111
P2	Y	0A0H	1111 1111
P3	Y	0B0H	1111 1111
IP	Y	0B8H	XX0 0000
IE	Y	0A8H	0XX0 0000
TMOD	N	89H	0000 0000
TCON	Y	88H	0000 0000
TH0	N	8CH	0000 0000
TL0	N	8AH	0000 0000
TH1	N	8DH	0000 0000
TL1	N	8BH	0000 0000
SCON	Y	98H	0000 0000
SBUF	N	99H	Indeterminate
PCON	N	87H	HMOS 0XXX XXXX CHMOS 0XXX 0000

**8051 Clock and Instruction Cycle:**

In 8051, one instruction cycle consists of twelve (12) clock cycles. Instruction cycle is sometimes called as Machine cycle by some authors.



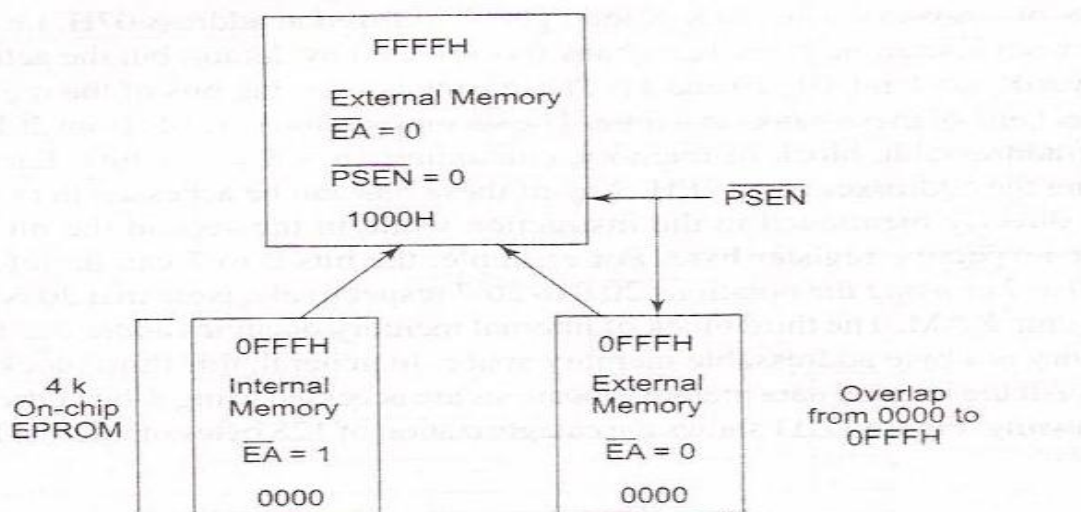
**Fig 5.2 : Instruction cycle of 8051**

In 8051, each instruction cycle has six states ( $S_1 - S_6$ ). Each state has two pulses (P1 and P2)

✚ **MEMORY ADDRESSING**

- Program memory - EPROM
  - Intermediate results, variables, const
  - 4KB internal from 0000 – 0FFFH
  - 64KB external with PSEN, till FFFFH
  - Internal –external difference PSEN
- Data Memory – RAM
  - 64KB of external with DPTR signal
  - Internal memory two parts - 128 bytes Internal RAM and secondly set of addresses from 80-FFH for SFR's
  - 128 bytes from 00 – 7FH direct or indirect

SFR addresses – only direct addressing mode



\* On chip EPROM may be 8 k/16 k in some versions of 8051

**Fig. 17.9 Program Memory Map of an 8051 System**

- Lower 128 bytes in three sections
  - 00-1F – 32 bytes 4 banks 00,01,10,11 each containing 8 registers of 8 bits each. Only one accessible at a time with PSW bits.
  - 20-2FH – 16bytes is bit addressable with addresses 0F to 7FH, 20.7 or 20.0, or 0-7
  - 30-7F – 80 bytes of general purpose data memory. It is byte addressable, used for stack

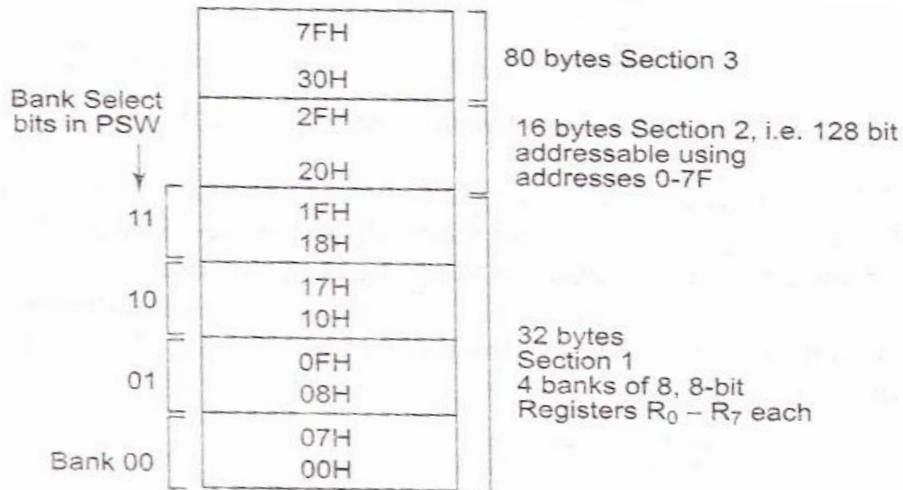
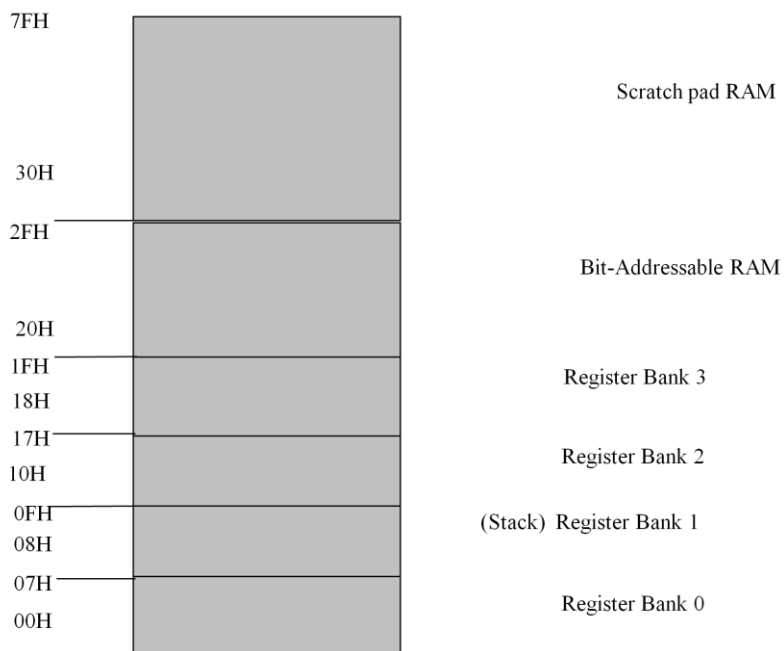
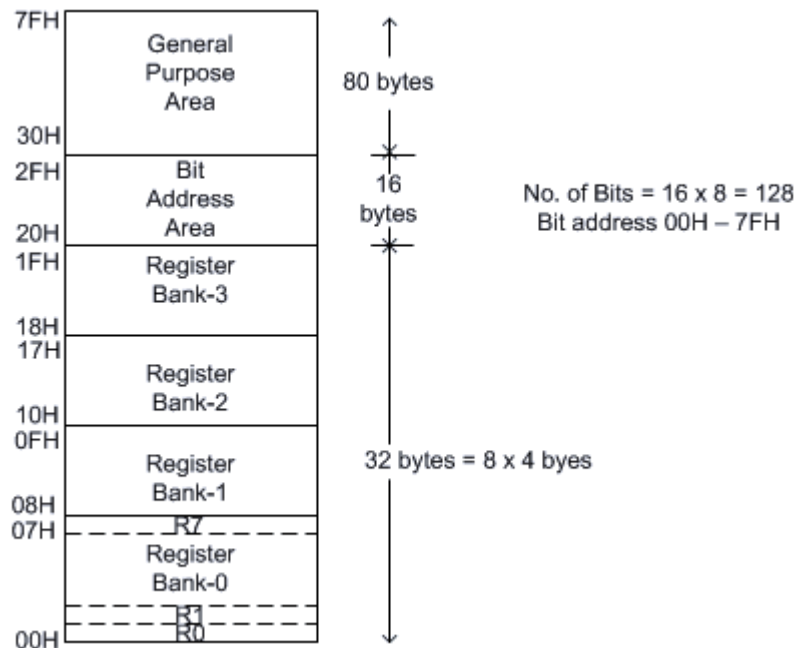


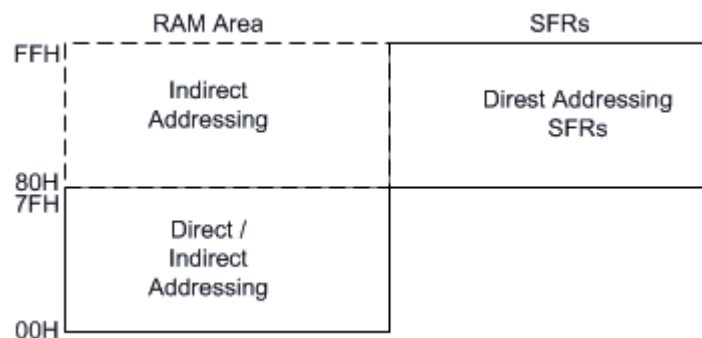
Fig. 17.11 Functional Description of Internal Lower 128 Bytes of RAM

• RAM memory space allocation in the 8051



**128 bytes of Internal RAM Structure (lower address space)****Fig: Internal RAM Structure**

The lower 32 bytes are divided into 4 separate banks. Each register bank has 8 registers of one byte each. A register bank is selected depending upon two bank select bits in the PSW register. Next 16 bytes are bit addressable. In total, 128 bits (16 × 8) are available in bit addressable area. Each bit can be accessed and modified by suitable instructions. The bit addresses are from 00H (LSB of the first byte in 20H) to 7FH (MSB of the last byte in 2FH). Remaining 80 bytes of RAM are available for general purpose.

**Internal Data Memory and Special Function Register (SFR) Map****Fig: Internal Data Memory Map**

The special function registers (SFRs) are mapped in the upper 128 bytes of internal data memory address. Hence there is an address overlap between the upper 128 bytes of data RAM and SFRs. Please note that the upper 128 bytes of data RAM are present only in the 8052 family. The lower 128 bytes of RAM (00H - 7FH) can be accessed both by direct or indirect addressing while the upper 128 bytes of RAM (80H - FFH) are accessed by indirect addressing. The SFRs (80H - FFH) are accessed by direct addressing only. This feature distinguishes the upper 128 bytes of memory from the SFRs, as shown in figure above.

** TIMERS / COUNTERS**

8051 has two 16-bit programmable UP timers/counters. They can be configured to operate either as timers or as event counters. The names of the two counters are T0 and T1 respectively. The timer content is available in four 8-bit special function registers, viz, TLO, TH0, TL1 and TH1 respectively.

In the "timer" function mode, the counter is incremented in every machine cycle. Thus, one can think of it as counting machine cycles. Hence the clock rate is  $1/12^{th}$  of the oscillator frequency.

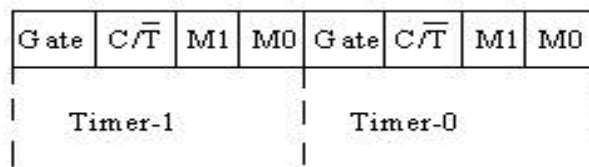
In the "counter" function mode, the register is incremented in response to a 1 to 0 transition at its corresponding external input pin (T0 or T1). It requires 2 machine cycles to detect a high to low transition. Hence maximum count rate is  $1/24^{th}$  of oscillator frequency.

The operation of the timers/counters is controlled by two special function registers, TMOD and TCON respectively.

**Timer Mode control (TMOD) Special Function Register:**

TMOD register is not bit addressable.

TMOD Address: 89 H



Various bits of TMOD are described as follows -

**Gate:** This is an OR Gate enabled bit which controls the effect of  $\overline{INT1/0}$  on START/STOP of Timer. It is set to one ('1') by the program to enable the interrupt to start/stop the timer. If TR1/0 in TCON is set and signal on  $\overline{INT1/0}$  pin is high then the timer starts counting using either internal clock (timer mode) or external pulses (counter mode).

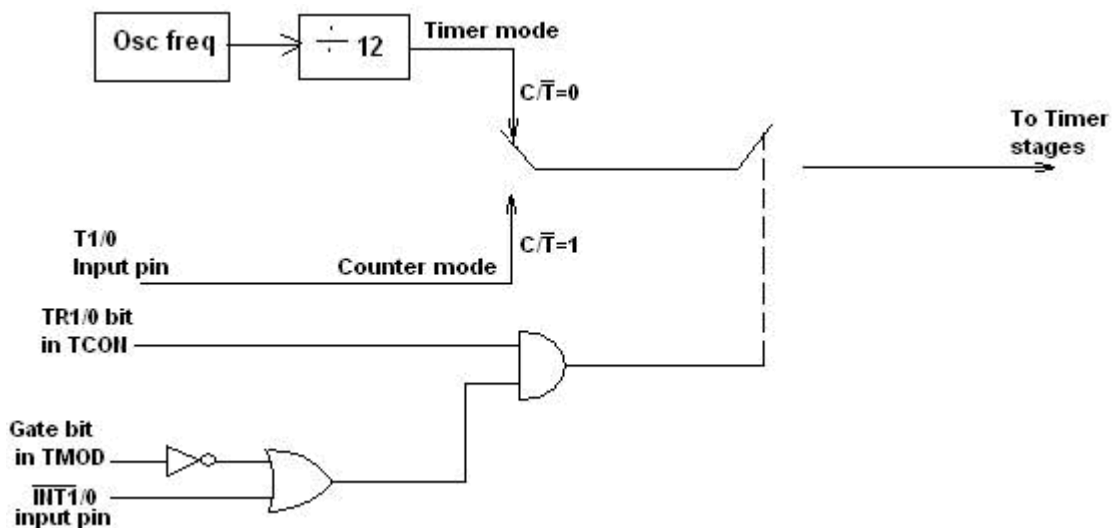
**C/ $\bar{T}$ :** It is used for the selection of Counter/Timer mode.

Mode Select Bits:

M1	M0	Mode
0	0	Mode 0
0	1	Mode 1
1	0	Mode 2
1	1	Mode 3

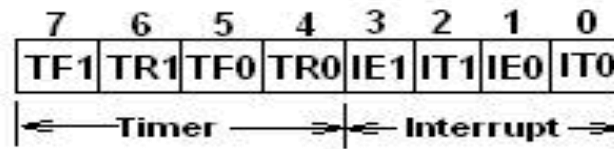
M1 and M0 are mode select bits.

**Timer/ Counter control logic:**



**Timer control (TCON) Special function register:**

TCON is bit addressable. The address of TCON is 88H. It is partly related to Timer and partly to interrupt.



The various bits of TCON are as follows.

**TF1:** Timer1 overflow flag. It is set when timer rolls from all 1s to 0s. It is cleared when processor vectors to execute ISR located at address 001BH.

**TR1:** Timer1 run control bit. Set to 1 to start the timer / counter.

**TF0:** Timer0 overflow flag. (Similar to TF1)

**TR0:** Timer0 run control bit.

**IE1:** Interrupt1 edge flag. Set by hardware when an external interrupt edge is detected. It is cleared when interrupt is processed.

**IE0:** Interrupt0 edge flag. (Similar to IE1)

**IT1:** Interrupt1 type control bit. Set/ cleared by software to specify falling edge / low level triggered external interrupt.

**IT0:** Interrupt0 type control bit. (Similar to IT1)

As mentioned earlier, Timers can operate in four different modes. They are as follows

## ✚ INTERRUPTS

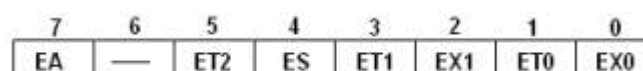
8051 provides 5 vectored interrupts. They are -

1.  $\overline{INT0}$
2. TF0
3.  $\overline{INT1}$
4. TF1
5. RI/TI

Out of these,  $\overline{INT0}$  and  $\overline{INT1}$  are external interrupts whereas Timer and Serial port interrupts are generated internally. The external interrupts could be negative edge triggered or low level triggered. All these interrupt, when activated, set the corresponding interrupt flags. Except for serial interrupt, the interrupt flags are cleared when the processor branches to the Interrupt Service Routine (ISR). The external interrupt flags are cleared on branching to Interrupt Service Routine (ISR), provided the interrupt is negative edge triggered. For low level triggered external interrupt as well as for serial interrupt, the corresponding flags have to be cleared by software by the programmer.

**Interrupt Enable register (IE):**

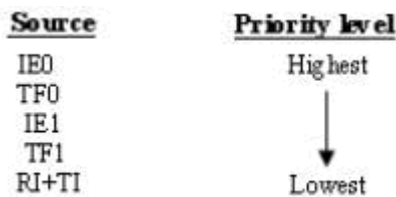
Address: A8H



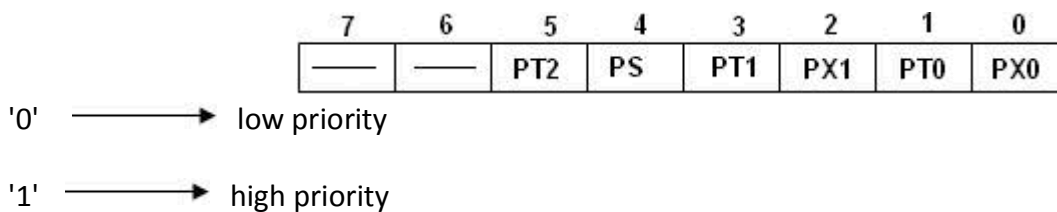
- EX0 →  $\overline{INT0}$  interrupt (External) enable bit
- ET0 → Timer-0 interrupt enable bit
- EX1 →  $\overline{INT1}$  interrupt (External) enable bit
- ET1 → Timer-1 interrupt enable bit
- ES → Serial port interrupt enable bit
- ET2 → Timer-2 interrupt enable bit
- EA → Enable/Disable all
- Setting '1' → Enable the corresponding interrupt
- Setting '0' → Disable the corresponding interrupt

**Priority level structure:**

Each interrupt source can be programmed to have one of the two priority levels by setting (high priority) or clearing (low priority) a bit in the IP (Interrupt Priority) Register . A low priority interrupt can itself be interrupted by a high priority interrupt, but not by another low priority interrupt. If two interrupts of different priority levels are received simultaneously, the request of higher priority level is served. If the requests of the same priority level are received simultaneously, an internal polling sequence determines which request is to be serviced. Thus, within each priority level, there is a second priority level determined by the polling sequence, as follows.



**Interrupt Priority register (IP)**



**SERIAL INTERFACE**

The serial port of 8051 is full duplex, i.e., it can transmit and receive simultaneously.

The register SBUF is used to hold the data. The special function register SBUF is physically two registers. One is, write-only and is used to hold data to be transmitted out of the 8051 via TXD. The other is, read-only and holds the received data from external sources via RXD. Both mutually exclusive registers have the same address 099H.

**Serial Port Control Register (SCON):**

Register SCON controls serial data communication.

Address: 098H (Bit addressable)

<b>SM0</b>	<b>SM1</b>	<b>SM2</b>	<b>REN</b>	<b>TB8</b>	<b>RB8</b>	<b>TI</b>	<b>RI</b>
------------	------------	------------	------------	------------	------------	-----------	-----------

Mode select bits

<b>SM0</b>	<b>SM1</b>	<b>Mode</b>
0	0	Mode 0
0	1	Mode 1
1	0	Mode 2
1	1	Mode 3

SM2: multi processor communication bit

REN: Receive enable bit

TB8: Transmitted bit 8 (Normally we have 0-7 bits transmitted/received)

RB8: Received bit 8

TI: Transmit interrupt flag

RI: Receive interrupt flag

### **Power Mode control Register(PCON):**

Register PCON controls processor power down, sleep modes and serial data baud rate. Only one bit of PCON is used with respect to serial communication. The seventh bit (b7) (SMOD) is used to generate the baud rate of serial communication.

Address: 87H

b7							b0
<b>SMOD</b>	—	—	—	<b>GF1</b>	<b>GF0</b>	<b>PD</b>	<b>IDL</b>

SMOD: Serial baud rate modify bit

GF1: General purpose user flag bit 1

GF0: General purpose user flag bit 0

PD: Power down bit

IDL: Idle mode bit

### **Data Transmission**

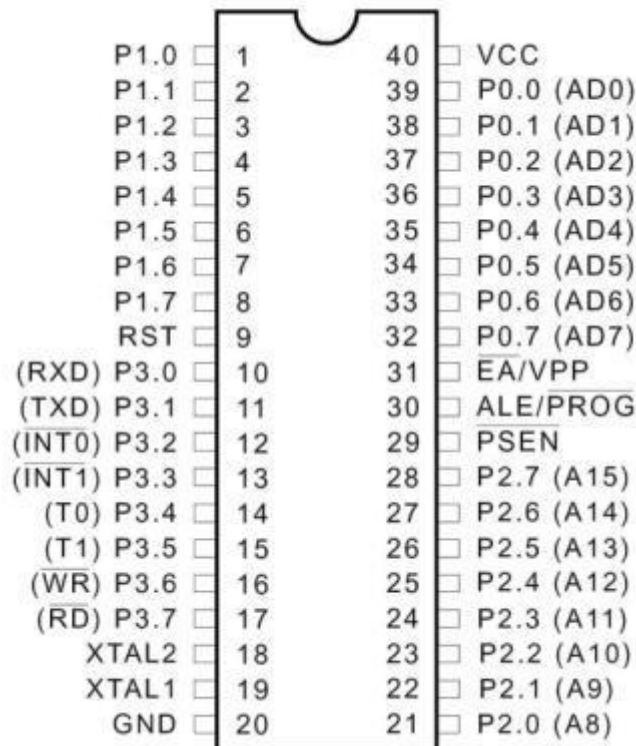
Transmission of serial data begins at any time when data is written to SBUF. Pin P3.1 (Alternate function bit TXD) is used to transmit data to the serial data network. TI is set to 1 when data has been transmitted. This signifies that SBUF is empty so that another byte can be sent.

### **Data Reception**

Reception of serial data begins if the receive enable bit is set to 1 for all modes. Pin P3.0 (Alternate function bit RXD) is used to receive data from the serial data network. Receive interrupt flag, RI, is set after the data has been received in all modes. The data gets stored in SBUF register from where it can be read.



## PIN DIAGRAM OF 8051



### Pin out Description:

**Pins 1-8: Port 1** Each of these pins can be configured as an input or an output.

**Pin 9: RS** A logic one on this pin disables the microcontroller and clears the contents of most registers. In other words, the positive voltage on this pin resets the microcontroller. By applying logic zero to this pin, the program starts execution from the beginning.

**Pins10-17: Port 3** Similar to port 1, each of these pins can serve as general input or output. Besides, all of them have alternative functions:

**Pin 10: RXD** Serial asynchronous communication input or Serial synchronous communication output.

**Pin 11: TXD** Serial asynchronous communication output or Serial synchronous communication clock output.

**Pin 12: INT0** Interrupt 0 input.

**Pin 13: INT1** Interrupt 1 input.

**Pin 14: T0** Counter 0 clock input.

**Pin 15: T1** Counter 1 clock input.

**Pin 16: WR** Write to external (additional) RAM.

**Pin 17: RD** Read from external RAM.

**Pin 18, 19: X2, X1** Internal oscillator input and output. A quartz crystal which specifies operating frequency is usually connected to these pins. Instead of it, miniature ceramics resonators can also be used for frequency stability. Later versions of microcontrollers operate at a frequency of 0 Hz up to over 50 Hz.

**Pin 20: GND** Ground.

**Pin 21-28: Port 2** If there is no intention to use external memory then these port pins are configured as general inputs/outputs. In case external memory is used, the higher address byte, i.e. addresses A8-A15 will appear on this port. Even though memory with capacity of 64Kb is not used, which means that not all eight port bits are used for its addressing, the rest of them are not available as inputs/outputs.

**Pin 29: PSEN** If external ROM is used for storing program then a logic zero (0) appears on it every time the microcontroller reads a byte from memory.

**Pin 30: ALE** Prior to reading from external memory, the microcontroller puts the lower address byte (A0-A7) on P0 and activates the ALE output. After receiving signal from the ALE pin, the external register (usually 74HCT373 or 74HCT375 add-on chip) memorizes the state of P0 and uses it as a memory chip address. Immediately after that, the ALU pin is returned its previous logic state and P0 is now used as a Data Bus. As seen, port data multiplexing is performed by means of only one additional (and cheap) integrated circuit. In other words, this port is used for both data and address transmission.

**Pin 31: EA** By applying logic zero to this pin, P2 and P3 are used for data and address transmission with no regard to whether there is internal memory or not. It means that even there is a program written to the microcontroller, it will not be executed. Instead, the program written to external ROM will be executed. By applying logic one to the EA pin, the microcontroller will use both memories, first internal then external (if exists).

**Pin 32-39: Port 0** Similar to P2, if external memory is not used, these pins can be used as general inputs/outputs. Otherwise, P0 is configured as address output (A0-A7) when the ALE pin is driven high (1) or as data output (Data Bus) when the ALE pin is driven low (0).

**Pin 40: VCC** +5V power supply.

## 8051 ADDRESSING MODES

8051 has four addressing modes.

- Immediate Addressing
- Bank Addressing or Register Addressing
- Direct Addressing
- Register Indirect Addressing

**Immediate Addressing:** Data is immediately available in the instruction.

For example -

ADD A, #77; Adds 77 (decimal) to A and stores in A

ADD A, #4DH; Adds 4D (hexadecimal) to A and stores in A

MOV DPTR, #1000H; Moves 1000 (hexadecimal) to data pointer

**Bank Addressing or Register Addressing:** This way of addressing accesses the bytes in the current register bank. Data is available in the register specified in the instruction. The register bank is decided by 2 bits of Processor Status Word (PSW).

For example-

ADD A, R0; Adds content of R0 to A and stores in A

**Direct Addressing:** The address of the data is available in the instruction.

For example -

MOV A, 088H; Moves content of SFR TCON (address 088H) to A

**Register Indirect Addressing:** The address of data is available in the R0 or R1 registers as specified in the instruction.

For example -

MOV A, @R0 moves content of address pointed by R0 to A

**External Data Addressing:** Pointer used for external data addressing can be either R0/R1 (256 byte access) or DPTR (64kbyte access).

For example -

MOVX A, @R0; Moves content of 8-bit address pointed by R0 to A

MOVX A, @DPTR; Moves content of 16-bit address pointed by DPTR to A

**External Code Addressing:** Sometimes we may want to store non-volatile data into the ROM e.g. look-up tables. Such data may require reading the code memory.

This may be done as follows -

MOVC A, @A+DPTR; Moves content of address pointed by A+DPTR to A

MOVC A, @A+PC; Moves content of address pointed by A+PC to A

## 8051 INSTRUCTIONS

8051 has about 111 instructions. These can be grouped into the following categories

1. Arithmetic Instructions
2. Logical Instructions
3. Data Transfer instructions
4. Boolean Variable Instructions
5. Program Branching Instructions

The following nomenclatures for register, data, address and variables are used while write instructions.

A: Accumulator

B: "B" register

C: Carry bit

Rn: Register R0 - R7 of the currently selected register bank

Direct: 8-bit internal direct address for data. The data could be in lower 128bytes of RAM (00 - 7FH) or it could be in the special function register (80 - FFH).

@Ri: 8-bit external or internal RAM address available in register R0 or R1. This is used for indirect addressing mode.

#data8: Immediate 8-bit data available in the instruction.

#data16: Immediate 16-bit data available in the instruction.

Addr11: 11-bit destination address for short absolute jump. Used by instructions AJMP & ACALL. Jump range is 2 kbyte (one page).

Addr16: 16-bit destination address for long call or long jump.

Rel: 2's complement 8-bit offset (one - byte) used for short jump (SJMP) and all conditional jumps.

bit: Directly addressed bit in internal RAM or SFR

### Arithmetic Instructions:

Mnemonics	Description	Bytes	Instruction Cycles
ADD A, Rn	$A \leftarrow A + Rn$	1	1
ADD A, direct	$A \leftarrow A + (\text{direct})$	2	1
ADD A, @Ri	$A \leftarrow A + @Ri$	1	1
ADD A, #data	$A \leftarrow A + \text{data}$	2	1
ADDC A, Rn	$A \leftarrow A + Rn + C$	1	1
ADDC A, direct	$A \leftarrow A + (\text{direct}) + C$	2	1
ADDC A, @Ri	$A \leftarrow A + @Ri + C$	1	1
ADDC A, #data	$A \leftarrow A + \text{data} + C$	2	1
DA A	Decimal adjust accumulator	1	1
DIV AB	Divide A by B $A \leftarrow \text{quotient}$ $B \leftarrow \text{remainder}$	1	4
DEC A	$A \leftarrow A - 1$	1	1
DEC Rn	$Rn \leftarrow Rn - 1$	1	1
DEC direct	$(\text{direct}) \leftarrow (\text{direct}) - 1$	2	1
DEC @Ri	$@Ri \leftarrow @Ri - 1$	1	1
INC A	$A \leftarrow A + 1$	1	1
INC Rn	$Rn \leftarrow Rn + 1$	1	1

INC direct	$(\text{direct}) \leftarrow (\text{direct}) + 1$	2	1
INC @Ri	$@Ri \leftarrow @Ri + 1$	1	1
INC DPTR	$DPTR \leftarrow DPTR + 1$	1	2
MUL AB	Multiply A by B $A \leftarrow \text{low byte } (A * B)$ $B \leftarrow \text{high byte } (A * B)$	1	4
SUBB A, Rn	$A \leftarrow A - Rn - C$	1	1
SUBB A, direct	$A \leftarrow A - (\text{direct}) - C$	2	1
SUBB A, @Ri	$A \leftarrow A - @Ri - C$	1	1
SUBB A, #data	$A \leftarrow A - \text{data} - C$	2	1

**Logical Instructions:**

Mnemonics	Description	Bytes	Instruction Cycles
ANL A, Rn	$A \leftarrow A \text{ AND } Rn$	1	1
ANL A, direct	$A \leftarrow A \text{ AND } (\text{direct})$	2	1
ANL A, @Ri	$A \leftarrow A \text{ AND } @Ri$	1	1
ANL A, #data	$A \leftarrow A \text{ AND } \text{data}$	2	1
ANL direct, A	$(\text{direct}) \leftarrow (\text{direct}) \text{ AND } A$	2	1
ANL direct, #data	$(\text{direct}) \leftarrow (\text{direct}) \text{ AND } \text{data}$	3	2
CLR A	$A \leftarrow 00H$	1	1
CPL A	$A \leftarrow \bar{A}$	1	1
ORL A, Rn	$A \leftarrow A \text{ OR } Rn$	1	1
ORL A, direct	$A \leftarrow A \text{ OR } (\text{direct})$	1	1
ORL A, @Ri	$A \leftarrow A \text{ OR } @Ri$	2	1
ORL A, #data	$A \leftarrow A \text{ OR } \text{data}$	1	1
ORL direct, A	$(\text{direct}) \leftarrow (\text{direct}) \text{ OR } A$	2	1
ORL direct, #data	$(\text{direct}) \leftarrow (\text{direct}) \text{ OR } \text{data}$	3	2
RL A	Rotate accumulator left	1	1
RLC A	Rotate accumulator left through carry	1	1
RR A	Rotate accumulator right	1	1
RRC A	Rotate accumulator right through carry	1	1
SWAP A	Swap nibbles within Accumulator	1	1
XRL A, Rn	$A \leftarrow A \text{ EXOR } Rn$	1	1
XRL A, direct	$A \leftarrow A \text{ EXOR } (\text{direct})$	1	1
XRL A, @Ri	$A \leftarrow A \text{ EXOR } @Ri$	2	1
XRL A, #data	$A \leftarrow A \text{ EXOR } \text{data}$	1	1
XRL direct, A	$(\text{direct}) \leftarrow (\text{direct}) \text{ EXOR } A$	2	1
XRL direct, #data	$(\text{direct}) \leftarrow (\text{direct}) \text{ EXOR } \text{data}$	3	2

**Data Transfer Instructions:**

Mnemonics	Description	Bytes	Instruction Cycles
MOV A, Rn	$A \leftarrow Rn$	1	1
MOV A, direct	$A \leftarrow (\text{direct})$	2	1

MOV A, @Ri	$A \leftarrow @Ri$	1	1
MOV A, #data	$A \leftarrow \text{data}$	2	1
MOV Rn, A	$Rn \leftarrow A$	1	1
MOV Rn, direct	$Rn \leftarrow (\text{direct})$	2	2
MOV Rn, #data	$Rn \leftarrow \text{data}$	2	1
MOV direct, A	$(\text{direct}) \leftarrow A$	2	1
MOV direct, Rn	$(\text{direct}) \leftarrow Rn$	2	2
MOV direct1, direct2	$(\text{direct1}) \leftarrow (\text{direct2})$	3	2
MOV direct, @Ri	$(\text{direct}) \leftarrow @Ri$	2	2
MOV direct, #data	$(\text{direct}) \leftarrow \text{#data}$	3	2
MOV @Ri, A	$@Ri \leftarrow A$	1	1
MOV @Ri, direct	$@Ri \leftarrow (\text{direct})$	2	2
MOV @Ri, #data	$@Ri \leftarrow \text{data}$	2	1
MOV DPTR, #data16	$DPTR \leftarrow \text{data16}$	3	2
MOVC A, @A+DPTR	$A \leftarrow \text{Code byte pointed by } A + DPTR$	1	2
MOVC A, @A+PC	$A \leftarrow \text{Code byte pointed by } A + PC$	1	2
MOVC A, @Ri	$A \leftarrow \text{Code byte pointed by Ri 8-bit address}$	1	2
MOVX A, @DPTR	$A \leftarrow \text{External data pointed by DPTR}$	1	2
MOVX @Ri, A	$@Ri \leftarrow A$ (External data - 8bit address)	1	2
MOVX @DPTR, A	$@DPTR \leftarrow A$ (External data - 16bit address)	1	2
PUSH direct	$(SP) \leftarrow (\text{direct})$	2	2
POP direct	$(\text{direct}) \leftarrow (SP)$	2	2
XCH Rn	Exchange A with Rn	1	1
XCH direct	Exchange A with direct byte	2	1
XCH @Ri	Exchange A with indirect RAM	1	1
XCHD A, @Ri	Exchange least significant nibble of A with that of indirect RAM	1	1

### Boolean Variable Instructions:

Mnemonics	Description	Bytes	Instruction Cycles
CLR C	$C\text{-bit} \leftarrow 0$	1	1
CLR bit	$\text{bit} \leftarrow 0$	2	1
SET C	$C \leftarrow 1$	1	1
SET bit	$\text{bit} \leftarrow 1$	2	1
CPL C	$C \leftarrow \overline{C\text{-bit}}$	1	1
CPL bit	$\text{bit} \leftarrow \overline{\text{bit}}$	2	1
ANL C, /bit	$C \leftarrow C \cdot \overline{\text{bit}}$	2	1
ANL C, bit	$C \leftarrow C \cdot \text{bit}$	2	1
ORL C, /bit	$C \leftarrow C + \overline{\text{bit}}$	2	1
ORL C, bit	$C \leftarrow C + \text{bit}$	2	1
MOV C, bit	$C \leftarrow \text{bit}$	2	1
MOV bit, C	$\text{bit} \leftarrow C$	2	2

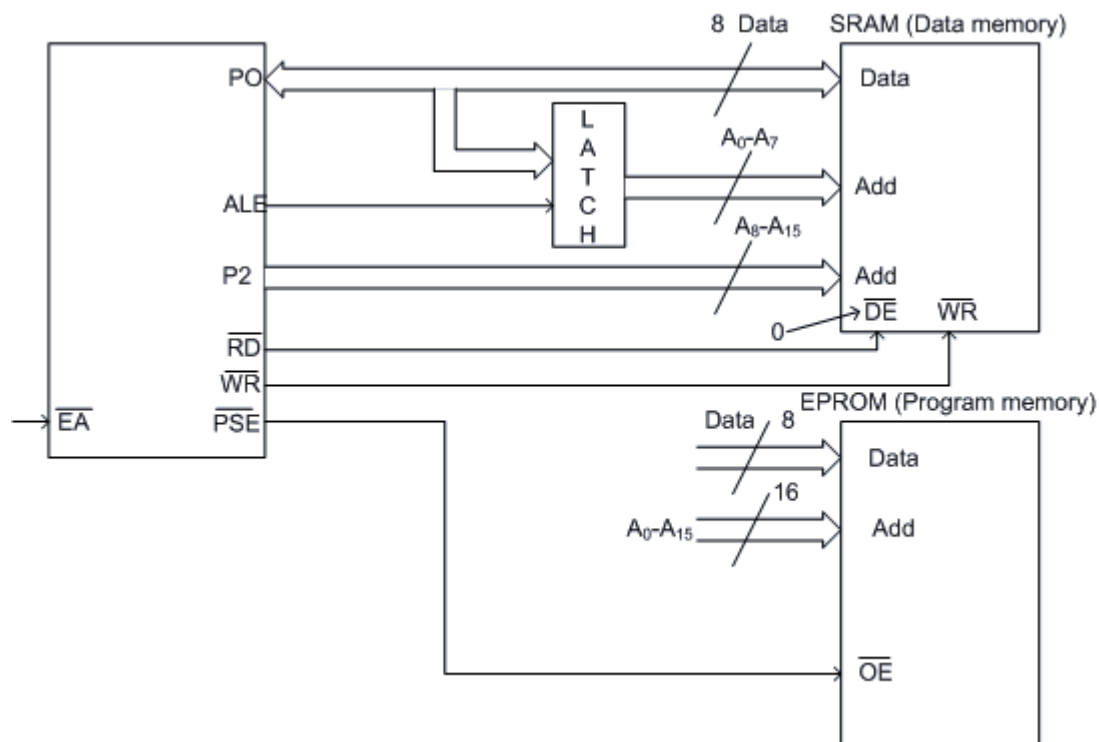
**Program Branching Instructions:**

Mnemonics	Description	Bytes	Instruction Cycles
ACALL addr11	PC + 2 $\rightarrow$ (SP) ; addr 11 $\rightarrow$ PC	2	2
AJMP addr11	Addr11 $\rightarrow$ PC	2	2
CJNE A, direct, rel	Compare with A, jump (PC + rel) if not equal	3	2
CJNE A, #data, rel	Compare with A, jump (PC + rel) if not equal	3	2
CJNE Rn, #data, rel	Compare with Rn, jump (PC + rel) if not equal	3	2
CJNE @Ri, #data, rel	Compare with @Ri A, jump (PC + rel) if not equal	3	2
DJNZ Rn, rel	Decrement Rn, jump if not zero	2	2
DJNZ direct, rel	Decrement (direct), jump if not zero	3	2
JC rel	Jump (PC + rel) if C bit = 1	2	2
JNC rel	Jump (PC + rel) if C bit = 0	2	2
JB bit, rel	Jump (PC + rel) if bit = 1	3	2
JNB bit, rel	Jump (PC + rel) if bit = 0	3	2
JBC bit, rel	Jump (PC + rel) if bit = 1	3	2
JMP @A+DPTR	A+DPTR $\rightarrow$ PC	1	2
JZ rel	If A=0, jump to PC + rel	2	2
JNZ rel	If A $\neq$ 0 , jump to PC + rel	2	2
LCALL addr16	PC + 3 $\rightarrow$ (SP), addr16 $\rightarrow$ PC	3	2
LJMP addr 16	Addr16 $\rightarrow$ PC	3	2
NOP	No operation	1	1
RET	(SP) $\rightarrow$ PC	1	2
RETI	(SP) $\rightarrow$ PC, Enable Interrupt	1	2
SJMP rel	PC + 2 + rel $\rightarrow$ PC	2	2
JMP @A+DPTR	A+DPTR $\rightarrow$ PC	1	2
JZ rel	If A = 0. jump PC+ rel	2	2
JNZ rel	If A $\neq$ 0, jump PC + rel	2	2
NOP	No operation	1	1

## EXTERNAL MEMORY AND I/O INTERFACING

### Interfacing External Memory:

If external program/data memory are to be interfaced, they are interfaced in the following way.



**Fig 6.1: Circuit Diagram for Interfacing of External Memory**

External program memory is fetched if either of the following two conditions are satisfied.

1.  $\overline{EA}$  (Enable Address) is low. The microcontroller by default starts searching for program from external program memory.
2. PC is higher than FFFH for 8051 or 1FFFH for 8052.

$\overline{PSEN}$  tells the outside world whether the external memory fetched is program memory or data memory.  $\overline{EA}$  is user configurable.  $\overline{PSEN}$  is processor controlled.

## INTRODUCTION TO ASSEMBLY LEVEL LANGUAGE

**[Label:] mnemonic [operands] ;comment**

Mnemonics → assembly level instructions are called mnemonic like MOV R5

Operands → on which the operation is performed.

### Example:

Loop: MOV R1, #25H; transfer 25H into R1

↑    ↑    ↑                    ↑  
Label mnemonics operand comments

→ two instructions which are used to start and terminate program.

• **ORG** → this instruction indicate the origin of program ORG 3000H

→ means program starts from 3000H locn.

→ this instruction hasn't take any memory space. It is used to show the



starting address of program.

- **END** → this instruction show the END of program or it is used to terminate the program.

**Example:**

ORG 0H;

Again: MOV R5, # 25H; transistor 25H to R5

ADD A, R5; Add the R5 with Acc

SJMP Again;

END;