8051 Microcontroller

Microprocessor is a single chip CPU, **microcontroller** contains, a CPU and much of the remaining circuitry of a complete microcomputer system in a single chip. **Microcontroller** includes RAM, ROM, serial and parallel interface, timer, interrupt schedule circuitry (in addition to CPU) in a single chip.

1) Computer System Vs Embedded System:

Microprocessor widely used in the computer system. And microcontroller is used in embedded system.

If the microprocessor is the heart of computer system then microcontroller is the heart of the embedded system.

2) Architecture:

The microprocessor uses **Von Neumann architecture** where data and program present in the same memory module .The microcontroller uses **Harvard architecture**. In this module, data and program get stored in separate memory. The microcontroller can access data and program at the same time as it is in a separate memory. This is one of the reasons microcontrollers is faster than the

microprocessor.

3) Memory and I/O Components:

The microprocessor can not operate without peripheral components. It has only processing unit and have to attach all the required components externally to operate. Whereas micro control has small processing unit along with internal memory to store and I/O components to give input. So it can work independently.

4) Circuit Size and its Complexity:

As we have to connect components externally, microprocessor circuit becomes large and complex. In microcontroller all the components are internally connected, its circuit becomes too small.

5) Efficient Techniques to use in Compact System:

The microcontroller can be used in the compact system as it has a small size. So microcontroller is better and efficient technique in the compact system than the microprocessor.

6) Cost:

Microprocessor requires external components to operate. So the Cost of the microprocessor is higher than the microcontroller.

7) Power consumption:

Microprocessor requires external components and its circuits also complex. It requires more power consumption. So it is difficult to operate microprocessor using battery power. The microcontroller has very low external components. it manages all its operation inside the single chip. So it consumes very low power supply as compared to the microprocessor. We can operate microcontroller on the externally connected stored power such as a battery.

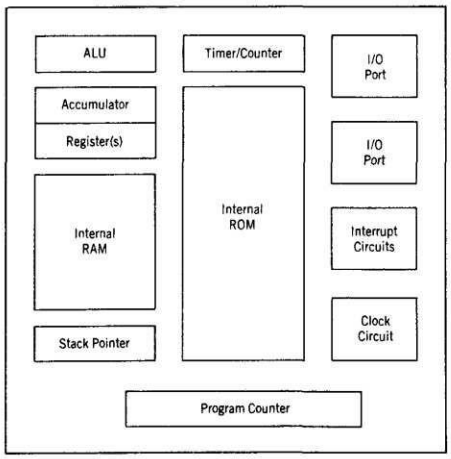
8) Power saving feature:

The microcontroller can have multiple modes of operation such as higher performance, balance, idle or power saving mode. So if we operate microcontroller in power saving mode, the power consumption re reduce even more. Most of the Microprocessor does not have this power saving feature.

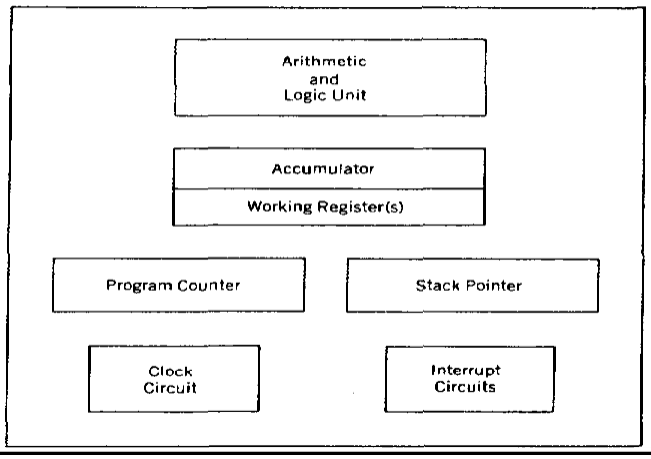
9) ProcessngSpeed:

The microprocessor has very less internal registers. It has to rely on external storage. So all the memory operations are carried out using memory based external commands. Results in high processing time.The microcontroller have many registers for instruction execution. Fetching data and storing data require internal commands. So its execution and processing time are lower than the microprocessor.

A Block Diagram of a Microcontroller



A Block Diagram of a Microprocessor



1 Overview

- The Intel 8051 is a very popular general purpose microcontroller widely used for small scale embedded systems.
- The 8051 is an 8-bit microcontroller with 8 bit data bus and 16-bit address bus.
- The 16 bit address bus can address a 64K(216) byte code memory space and a separate 64K byte of data memory space.
- The 8051 has 4K on-chip read only code memory and 128 bytes of internal Random Access Memory (RAM)
- Besides internal RAM, the 8051 has various
  - Special Function Registers (SFR) such as the Accumulator, the B register, and many other control registers.
  - 34 8-bit general purpose registers in total. The ALU performs one 8-bit operation at a time. They are register A,B and 32 working registers
- Two 16 bit /Counter timers
- Three internal interrupts (one serial), 2 external interrupts.
- Four 8-bit I/O ports (3 of them are dual purposed). One of them used for serial port,Some 8051 chips come with UART for serial communication and ADC for analog to digital conversion

8051 Chip Pins

- 40 pins on the 8051 chip. Most of these pins are used to connect to I/O devices or external data and code memory.
- 4 I/O port take 32 pins(4 x 8 bits) plus a pair of XTALS pins for crystal clock
- A pair of Vcc and GND pins for power supply (the 8051 chip needs +5V 500mA to function properly)
- A pair of timer pins for timing controls, a group of pins (EA, ALE, PSEN) for internal and external data and code memory access controls
- A pair of Crystal clock pins(XTAL1,2)
- One Reset pin for reboot purpose

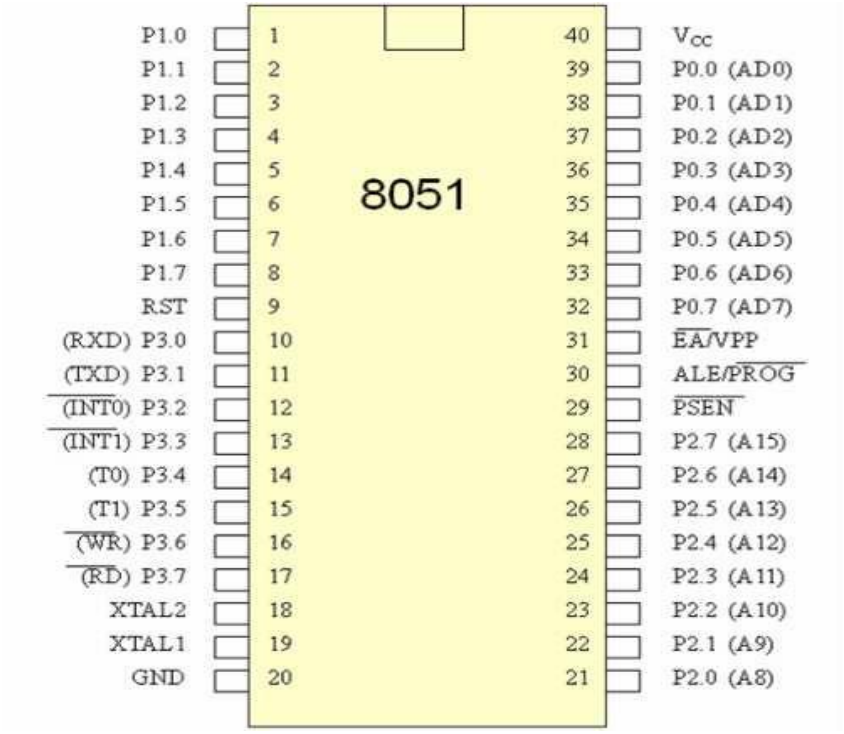


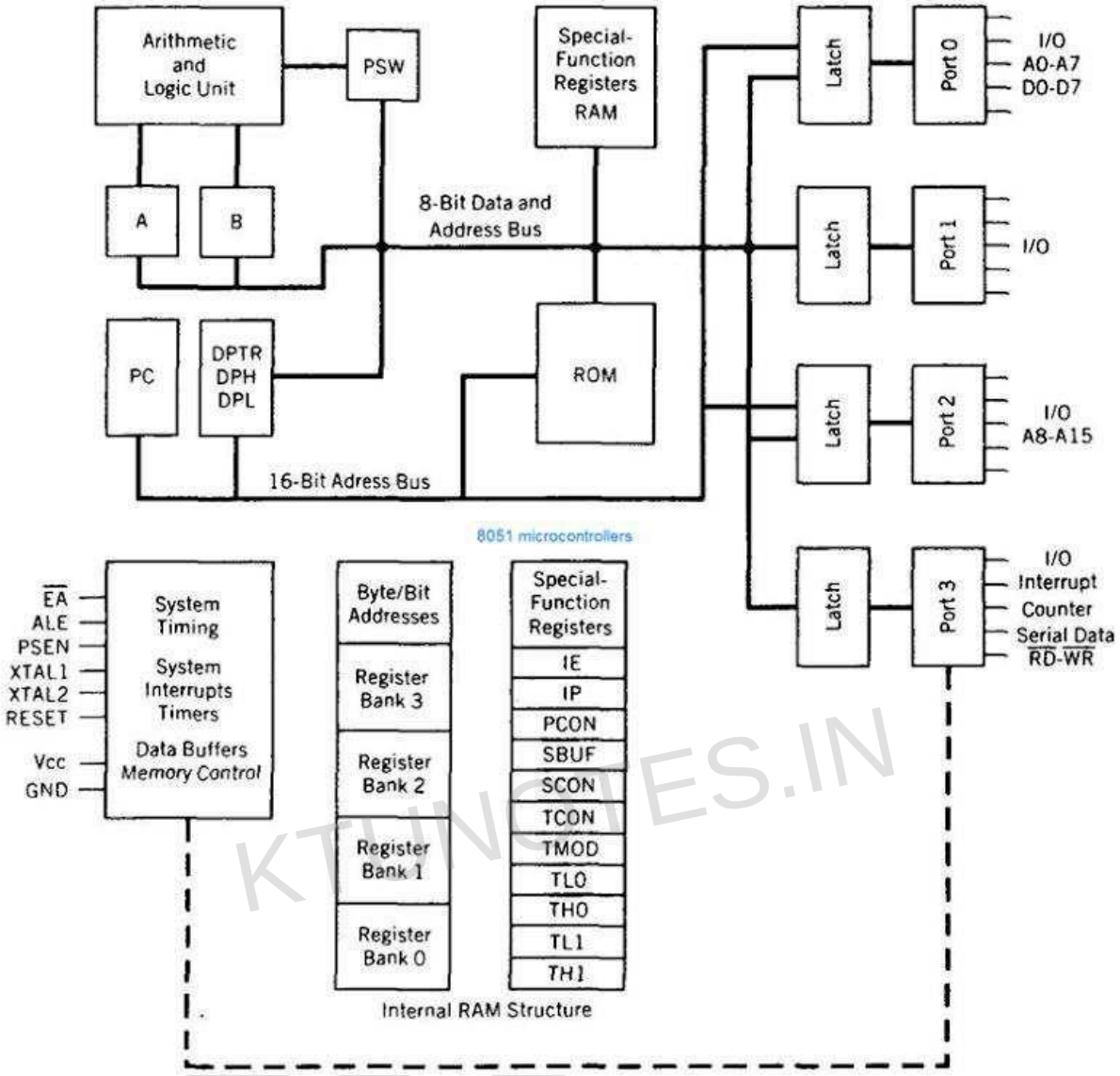
Fig Pin out Diagram of the 8051 Microcontroller

The Pin Connection for External Code and Data Memory

- The EA' (External Access) pin is used to control the internal or external memory access.
- The signal 0 is for external memory access and signal 1 for internal memory access.
- The PSEN' (Program Store Enable) is for reading external code memory when it is low (0) and EA is also 0.
- The ALE (Address Latch Enable) activates the port 0 joined with port 2 to provide 16 bit external address bus to access the external memory.
- The ALE multiplexes the P0: 1 for latching address on P0 as A0-A7 in the 16 bit address
- buss, 0 for latching P0 as data I/O.
- P0.x is named ADx because P0 is multiplexed for Address bus and Data bus at different clock time. WR' provides the signal to write external data memory RD' provides the signal to read external data and code memory.

8051 Internal Architecture

Fig 8051 Internal Architecture



System Clock and Oscillator Circuits

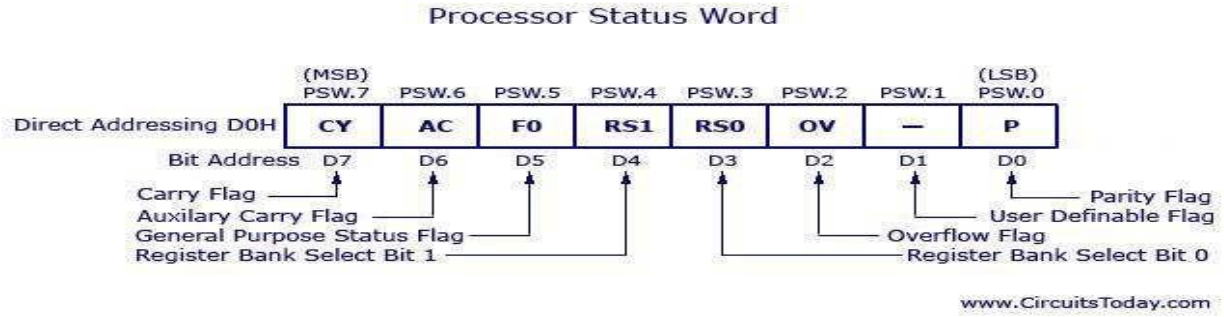
- The 8051 requires an external oscillator circuit. The oscillator circuit usually runs around 12MHz. The pulse is used to synchronize the system operation in a controlled pace.
- A machine cycle is minimum amount time a simplest machine instruction must take An 8051 machine cycle consists of 12 crystal pulses (ticks). The first 6 crystal pulses (clock cycle) is used to fetch the opcode and the second 6 pulses are used to perform the operation on the operands in the ALU.
- Pins used are
    - XTAL1
    - XTAL2

- The CPU has many important registers. The Program Count (PC) always holds the code memory location of next instruction. The CPU is the heart of any computer which is in charge of computer operations.
- It fetches instructions from the code memory into the instruction Register (IR), analyzes the opcode of the instruction, updates the PC to the location of next instruction, fetches the operand from the data memory if necessary, and finally performs the operation in the Arithmetic-Logic Unit (ALU) within the CPU.
- The B register is a register just for multiplication and division operation which requires more register spaces for the product of multiplication and the quotient and the remainder for the division. The immediate result is stored in the accumulator register (Acc) for next operation and the Program Status Word (PSW) is updated depending on the status of the operation result

.

Flags and PSW (Program Status Word) SFR for CPU status

Flags are one bit registers provided to store result of certain program instructions. Other instructions can test the conditions of the flags and make decisions based on the flag states. The flags are grouped into PSW and power control register (PCON).8051 has four math flags that respond automatically to output of math operations and three general purpose user flags can be set to 1 or 0 by the programmer. The math flags include Auxiliary carry(AC),Carry(C),Overflow(OV) and Parity(P).User flags are F0,GF0 and GF0F0 stored in PSW and GF0 & GF1 are stored in PCON



- P: parity check flag
- OV: ALU overflow flag
- RS0/RS1: Register bank specification mode
- 00: bank 0 (00H-07H); 01: bank1; 10: bank 2; 11: bank 3(18H- 1FH)
- F0: User defined lag
- CY: ALU carry out

- AC: ALU auxiliary carry out

Ports :

There are four 8-bit ports: P0, P1, P2 and P3. All of them are dual purpose ports except P1 which is only used for I/O.

- PORT P1 (Pins 1 to 8): The port P1 is a port dedicated for general I/O purpose. The other ports P0, P2 and P3 have dual roles in addition to their basic I/O function. PORT P0 (pins 32 to 39): When the external memory access is required then Port P0 is multiplexed for address bus and data bus that can be used to access external memory in conjunction with port P2. P0 acts as A0-A7 in address bus and D0-D7 for port data. It can be used for general purpose I/O if no external memory presents.
- PORT P2 (pins 21 to 28): Similar to P0, the port P2 can also play a role (A8-A15) in the address bus in conjunction with PORT P0 to access external memory.
- PORT P3 (Pins 10 to 17):
  - In addition to acting as a normal I/O port, • P3.0 can be used for serial receive input pin(RXD)
  - P3.1 can be used for serial transmit output pin(TXD) in a serial port
  - P3.2 and P3.3 can be used as external interrupt pins(INT0' and INT1')
  - P3.4 and P3.5 are used for external counter input pins(T0 and T1)
  - P3.6 and P3.7 can be used as external data memory write and read control signal pins(WR' and RD')read and write pins for memory access.

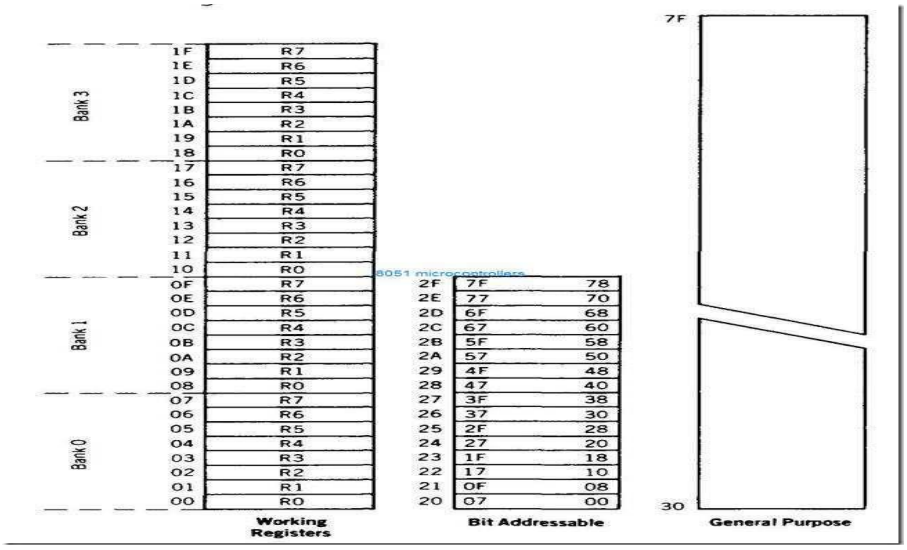
Internal Memory

The 8051 microcontroller contains internal RAM and internal ROM.RAM memory for variable data rhat can be altered as the program runs.

Internal RAM

The 128 byte internal RAM shown in below fig: is organized into three distinct areas

- The first 128 bytes of RAM (from 0000 to 0x7F) are called the direct memory, and can be used to store data.
- The lowest 32 bytes of RAM (00h to 1Fh) are reserved for 4 general register banks. The register banks numbered 0 to 3Each bank consist of 8 registers namely R0 to R7 with 8bit width and register can be addressed by its name or by its RAM address. The default bank is bank 0.



- A bit addressable area of 16 byte occupies RAM byte address 20h to 2Fh, forming total 128 byte addressable bit. An addressable bit may be specified by its bit address of 00h to 7Fh
- General purpose RAM area above the bit area from 30h to 7Fh.

Stack and Stack pointer

The stack rfers to an area in internal RAM to store and retrieve data quickly.The 8bit stack pointer (SP) register is used by the 8051 to hold internal RAM addressthat is called top of the stack.

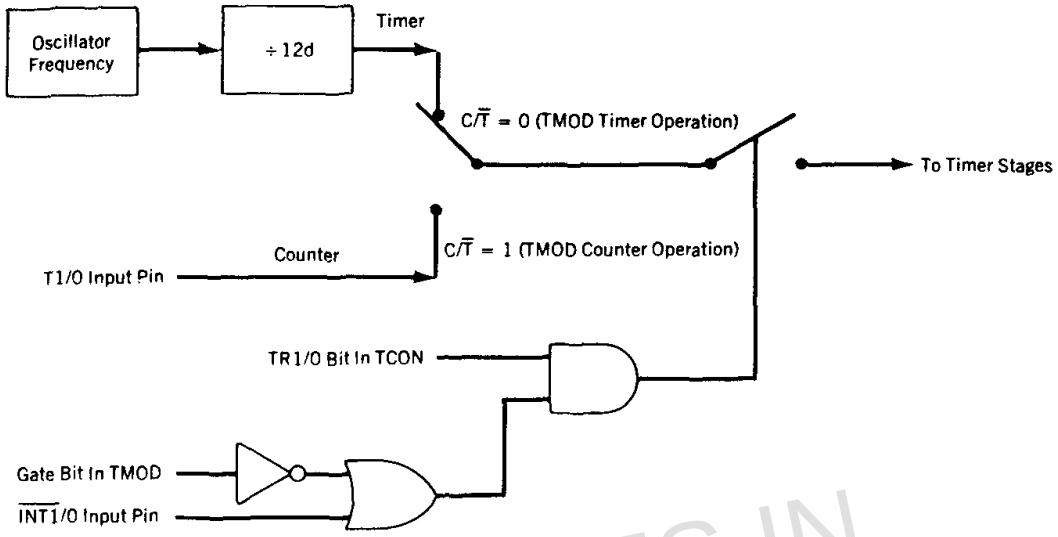
Special Function Registers (SFRs)

- The second 128 bytes are used to store Special Function Registers (SFR) that 8051 program can configure and control the ports, timer, interrupts, serial communication, and other tasks.
- The SFR is the upper area of addressable memory, from address 0x80 to 0xFF. This area consists of a series of memory-mapped ports and registers.
- All port input and output can therefore be performed by get and set operations on SFR port name such as P3.
- Also, different status registers are mapped into the SFR for checking the status of the 8051, and changing some operational parameters of the 8051.
- All 8051 CPU registers, I/O ports, timers and other architecture components are accessible in 8051 C through SFRs
- They are 21 SFRs and accessed in normal internal RAM (080H – 0FFH)

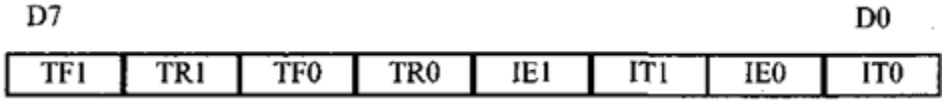
In addition to I/O ports, the most frequently used SFRs to control and configure 8051 operations are:TCON (Timer CONtrol),TMOD (Timer MODe),TH0/TH1 and TL0/TL1 (Timer's high and low bytes),SCON (Serial port CONtrol),IP (Interrupt Priority),IE (Interrupt Enable)

Counters and Timers

FIGURE 2.11 Timer/Counter Control Logic



Many microcontroller applications require the counting of external events such as frequency of a pulse train or generation of precise internal time delay between computer actions. For this purpose ,the 8051 contains two 16bit up counters T0 and T1.Each counter may be programmed to count internal clock pulses i.e. acting as timer or programmed to count external pulses as counter. The two counters are divided into two 8bit registers called timer low(TL0,TL1) and high(TH0,TH1) bytes. All counter actions are controlled by bits state in timer mode control register TMOD and timer control register TCON. Working as a "Timer", the timer is incremented by one every machine cycle. A machine cycle consists of 12 oscillator periods, so the count rate is 1/12 of the oscillator frequency. Working as a "Counter", the counter is incremented in response to a falling edge transition in the external input pins.



TCON (Timer/Counter) Register (Bit-addressable)

TF1 TCON.7 Timer 1 overflow flag. Set by hardware when timer/counter 1 overflows. Cleared by hardware as the processor vectors to the interrupt service routine.

TR1 TCON.6 Timer 1 run control bit. Set/cleared by software to turn timer/counter 1 on/off.

TF0 TCON.5 Timer 0 overflow flag. Set by hardware when timer/counter 0 overflows. Cleared by hardware as the processor vectors to the service routine.

TR0 TCON.4 Timer 0 run control bit. Set/cleared by software to turn timer/counter 0 on/off.

IE1 TCON.3 External interrupt 1 edge flag. Set by CPU when the external interrupt edge (H-to-L transition) is detected. Cleared by CPU when the interrupt is processed. *Note:* This flag does not latch low-level triggered interrupts.

IT1 TCON.2 Interrupt 1 type control bit. Set/cleared by software to specify falling edge/low-level triggered external interrupt.

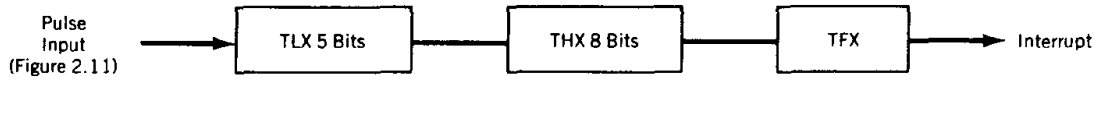
IE0 TCON.1 External interrupt 0 edge flag. Set by CPU when external interrupt (H-to-L transition) edge is detected. Cleared by CPU when interrupt is processed.

IT0 TCON.0 Interrupt 0 type control bit. Set/cleared by software to specify falling edge/low-level triggered external interrupt.

- Timer 0 and Timer 1 have four operating modes.

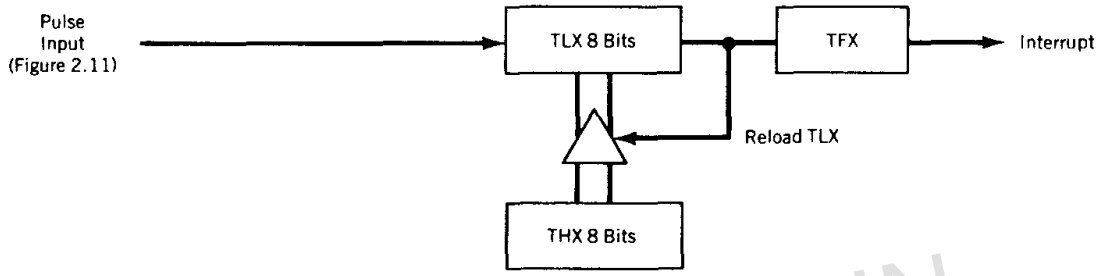
M1	M0	Mode Control
0	0	(Mode 0) 13 bit count mode
0	1	(Mode 1) 16 bit count mode
1	0	(Mode 2) Auto reload mode
1	1	(Mode 3) Multiple mode

FIGURE 2.12 Timer 1 and Timer 0 Operation Modes

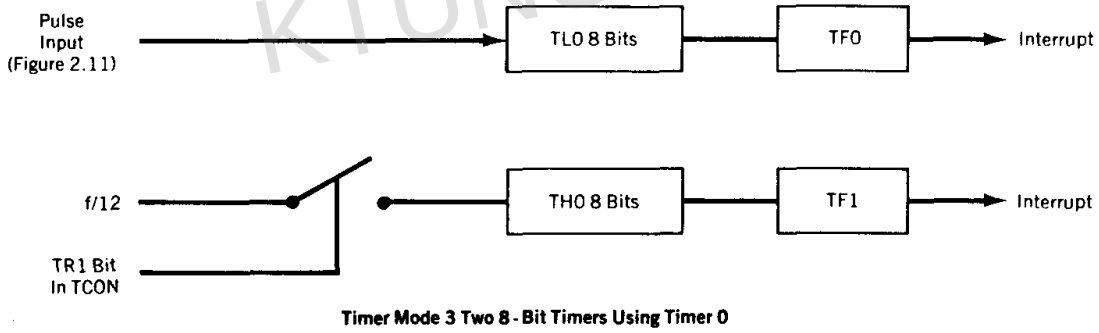


Timer Mode 0 13-Bit Timer/Counter

Timer Mode 1 16-Bit Timer/Counter



Timer Mode 2 Auto-Reload of TL from TH



Timer Mode 3 Two 8-Bit Timers Using Timer 0

Interrupts

- Interrupt is an internal or external event that suspends a program and transfers the control to an event handler or ISR to handle the event.
- After the service is over, the control is back to the suspended program to resume the execution.

IE enables interrupts

- EX0/EX1: Enable external interrupt INT0/INT1

- ET0/ET1: Enable Timer 0/Timer1 interrupt
- ES: Enable serial interrupt
- EA: Enable global interrupt
- Set 1 in a bit to enable its interrupt, e.g. EA =1;
- reset 0 to masks that interrupt, e.g., EA = 0;

Interrupt Flags

- The interrupt flags are set to 1 when the interrupts occur, the flag 0 indicates no interrupt.
- IE0/IE1 in TCON - For External Interrupts
- TF0/TF1 in TCON - For Timer Interrupts
- TI/RI in SCON - For Serial Interrupts

Interrupt Priority

- There are two types of interrupt priority
- User Defined Priority and Automatic Priority
- User Defined Priority
  - The IP register is used to define priority levels by users. The high priority interrupt can preempt the low priority interrupt. There are only two levels of interrupt priority.
- Automatic Priority
  - In each priority level, a priority is given in order of INT0, TF0, INT1, TF1, SI.
  - For example, if two external interrupts are set at same priority level, then INT0 has precedence over INT1.

1) External Interrupts

• An external interrupt is triggered by a low level or negative edge on INT0 and INT1 which depends on the external interrupt type setting. Set up an external interrupt type by IT0 and IT1 of TCON SFR.

2) Timer/Counter Interrupts

- This unit can be used as a counter to count external pulses on P3.4 and P3.5 pins or it can be used to count the pulses produced by the crystal oscillator of the microcontroller.
- Timer Interrupt is caused by Timer 0/ Timer1 overflow.

3)Serial Interrupts

- Serial communication with Universal Asynchronous Receive Transmit (UART) protocol transmits or receives the bits of a byte one after the other in a timed sequence on a single wire.
- It is used to communicate any serial port of devices and computers.
- The serial interrupt is caused by completion of a serial byte transmitting or receiving.
- The transmit data pin (TxD) is at P3.1 and the receive data pin (RxD) is at P3.0.
- All communication modes are controlled through SCON, a non bit addressable SFR. The SCON bits are defined as SM0, SM1, SM2, REN, TB8, RB8, TI, RI.

IE (Interrupt Enable Register) SFR used for interrupt control

- EX0/EX1 : Enables(1)/disables(0) the external interrupt 0
- and the external interrupt 1 on port P3.2 / P3.3
- ET0/ET1 : Enables(1)/disables(0) the Timer0 and Timer1
- interrupt via TF0/1
- ES : Enables(1)/disables(0) the serial port interrupt for sending and receiving data
- EA : Enables(1)/disables(0) all interrupts

IE : Interrupt Enable Register (Bit Addressable)

If the bit is 0, the corresponding interrupt is disabled. If the bit is 1, the corresponding interrupt is enabled.

EA	–	–	ES	ET1	EX1	ET0	EX0

EA	IE.7	Disables all interrupts. If EA = 0, no interrupt will be acknowledged. If EA = 1, interrupt source is individually enable or disabled by setting or clearing its enable bit.
-	IE.6	Not implemented, reserved for future use*.
-	IE.5	Not implemented, reserved for future use*.
ES	IE.4	Enable or disable the Serial port interrupt.
ET1	IE.3	Enable or disable the Timer 1 overflow interrupt.
EX1	IE.2	Enable or disable External interrupt 1.
ET0	IE.1	Enable or disable the Timer 0 overflow interrupt.
EX0	IE.0	Enable or disable External Interrupt 0.

IP (Interrupt Priority Register) SFR used for IP setting

- PX0/1: External interrupt 0/1 priority level

- PT0/1/2: Timer0, Timer1, Timer2(8052) interrupt priority level
- PS: Serial port interrupt priority level

Depending on operation they perform, all instructions are divided in several groups:

- Arithmetic Instructions
- Branch Instructions
- Data Transfer Instructions
- Logic Instructions
- Bit-oriented Instructions

The first part of each instruction, called MNEMONIC refers to the operation an instruction performs (copy, addition, logic operation etc.). Mnemonics are abbreviations of the name of operation being executed. For example:

- INC R1 – Means: Increment register R1 (increment register R1);
- LJMP LAB5 – Means: Long Jump LAB5 (long jump to the address marked as LAB5);
- JNZ LOOP – Means: Jump if Not Zero LOOP (if the number in the accumulator is not 0, jump to the address marked as LOOP);

The other part of instruction, called OPERAND is separated from mnemonic by at least one whitespace and defines data being processed by instructions. Some of the instructions have no operand, while some of them have one, two or three. If there is more than one operand in an instruction, they are separated by a comma. For example:

- RET – return from a subroutine;
- JZ TEMP – if the number in the accumulator is not 0, jump to the address marked as TEMP;
- ADD A,R3 – add R3 and accumulator;
- CJNE A,#20,LOOP – compare accumulator with 20. If they are not equal, jump to the address marked as LOOP;

Arithmetic instructions

Arithmetic instructions perform several basic operations such as addition, subtraction, division, multiplication etc. After execution, the result is stored in the first operand. For example:

ADD A,R1 – The result of addition (A+R1) will be stored in the accumulator.

ARITHMETIC INSTRUCTIONS

Mnemonic	Description	Byte	Cycle
ADD A,Rn	Adds the register to the accumulator	1	1
ADD A,direct	Adds the direct byte to the accumulator	2	2
ADD A,@Ri	Adds the indirect RAM to the accumulator	1	2
ADD A,#data	Adds the immediate data to the accumulator	2	2
ADDC A,Rn	Adds the register to the accumulator with a carry flag	1	1
ADDC A,direct	Adds the direct byte to the accumulator with a carry flag	2	2
ADDC A,@Ri	Adds the indirect RAM to the accumulator with a carry flag	1	2
ADDC A,#data	Adds the immediate data to the accumulator with a carry flag	2	2
SUBB A,Rn	Subtracts the register from the accumulator with a borrow	1	1
SUBB A,direct	Subtracts the direct byte from the accumulator with a borrow	2	2
SUBB A,@Ri	Subtracts the indirect RAM from the accumulator with a borrow	1	2
SUBB A,#data	Subtracts the immediate data from the accumulator with a borrow	2	2
INC A	Increments the accumulator by 1	1	1
INC Rn	Increments the register by 1	1	2
INC Rx	Increments the direct byte by 1	2	3
INC @Ri	Increments the indirect RAM by 1	1	3
DEC A	Decrements the accumulator by 1	1	1
DEC Rn	Decrements the register by 1	1	1
DEC Rx	Decrements the direct byte by 1	1	2
DEC @Ri	Decrements the indirect RAM by 1	2	3
INC DPTR	Increments the Data Pointer by 1	1	3
MUL AB	Multiplies A and B	1	5

DIV AB	Divides A by B	1	5
DA A	Decimal adjustment of the accumulator according to BCD code	1	1

Branch Instructions

There are two kinds of branch instructions:

- Unconditional jump instructions: upon their execution a jump to a new location from where the program continues execution is executed.
- Conditional jump instructions: a jump to a new program location is executed only if a specified condition is met. Otherwise, the program normally proceeds with the next instruction.

BRANCH INSTRUCTIONS

Mnemonic	Description	Byte	Cycle
ACALL addr11	Absolute subroutine call	2	6
LCALL addr16	Long subroutine call	3	6
RET	Returns from subroutine	1	4
RETI	Returns from interrupt subroutine	1	4
AJMP addr11	Absolute jump	2	3
LJMP addr16	Long jump	3	4
SJMP rel	Short jump (from –128 to +127 locations relative to the following instruction)	2	3
JC rel	Jump if carry flag is set. Short jump.	2	3
JNC rel	Jump if carry flag is not set. Short jump.	2	3
JB bit,rel	Jump if direct bit is set. Short jump.	3	4
JBC bit,rel	Jump if direct bit is set and clears bit. Short jump.	3	4
JMP @A+DPTR	Jump indirect relative to the DPTR	1	2
JZ rel	Jump if the accumulator is zero. Short jump.	2	3
JNZ rel	Jump if the accumulator is not zero. Short jump.	2	3

CJNE A,direct,rel	Compares direct byte to the accumulator and jumps if not equal. Short jump.	3	4
CJNE A,#data,rel	Compares immediate data to the accumulator and jumps if not equal. Short jump.	3	4
CJNE Rn,#data,rel	Compares immediate data to the register and jumps if not equal. Short jump.	3	4
CJNE @Ri,#data,rel	Compares immediate data to indirect register and jumps if not equal. Short jump.	3	4
DJNZ Rn,rel	Decrements register and jumps if not 0. Short jump.	2	3
DJNZ Rx,rel	Decrements direct byte and jump if not 0. Short jump.	3	4
NOP	No operation	1	1

Data Transfer Instructions

Data transfer instructions move the content of one register to another. The register the content of which is moved remains unchanged. If they have the suffix "X" (MOVX), the data is exchanged with external memory.

DATA TRANSFER INSTRUCTIONS

Mnemonic	Description	Byte	Cycle
MOV A,Rn	Moves the register to the accumulator	1	1
MOV A,direct	Moves the direct byte to the accumulator	2	2
MOV A,@Ri	Moves the indirect RAM to the accumulator	1	2
MOV A,#data	Moves the immediate data to the accumulator	2	2
MOV Rn,A	Moves the accumulator to the register	1	2
MOV Rn,direct	Moves the direct byte to the register	2	4
MOV Rn,#data	Moves the immediate data to the register	2	2
MOV direct,A	Moves the accumulator to the direct byte	2	3
MOV direct,Rn	Moves the register to the direct byte	2	3
MOV direct,direct	Moves the direct byte to the direct byte	3	4
MOV direct,@Ri	Moves the indirect RAM to the direct byte	2	4

MOV direct,#data	Moves the immediate data to the direct byte	3	3
MOV @Ri,A	Moves the accumulator to the indirect RAM	1	3
MOV @Ri,direct	Moves the direct byte to the indirect RAM	2	5
MOV @Ri,#data	Moves the immediate data to the indirect RAM	2	3
MOV DPTR,#data	Moves a 16-bit data to the data pointer	3	3
MOVC A,@A+DPTR	Moves the code byte relative to the DPTR to the accumulator (address=A+DPTR)	1	3
MOVC A,@A+PC	Moves the code byte relative to the PC to the accumulator (address=A+PC)	1	3
MOVX A,@Ri	Moves the external RAM (8-bit address) to the accumulator	1	3-10
MOVX A,@DPTR	Moves the external RAM (16-bit address) to the accumulator	1	3-10
MOVX @Ri,A	Moves the accumulator to the external RAM (8-bit address)	1	4-11
MOVX @DPTR,A	Moves the accumulator to the external RAM (16-bit address)	1	4-11
PUSH direct	Pushes the direct byte onto the stack	2	4
POP direct	Pops the direct byte from the stack/td>	2	3
XCH A,Rn	Exchanges the register with the accumulator	1	2
XCH A,direct	Exchanges the direct byte with the accumulator	2	3
XCH A,@Ri	Exchanges the indirect RAM with the accumulator	1	3
XCHD A,@Ri	Exchanges the low-order nibble indirect RAM with the accumulator	1	3

Logic Instructions

Logic instructions perform logic operations upon corresponding bits of two registers. After execution, the result is stored in the first operand.

Mnemonic	Description	Byte	Cycle
ANL A,Rn	AND register to accumulator	1	1
ANL A,direct	AND direct byte to accumulator	2	2
ANL A,@Ri	AND indirect RAM to accumulator	1	2

ANL A,#data	AND immediate data to accumulator	2	2
ANL direct,A	AND accumulator to direct byte	2	3
ANL direct,#data	AND immediae data to direct register	3	4
ORL A,Rn	OR register to accumulator	1	1
ORL A,direct	OR direct byte to accumulator	2	2
ORL A,@Ri	OR indirect RAM to accumulator	1	2
ORL direct,A	OR accumulator to direct byte	2	3
ORL direct,#data	OR immediate data to direct byte	3	4
XRL A,Rn	Exclusive OR register to accumulator	1	1
XRL A,direct	Exclusive OR direct byte to accumulator	2	2
XRL A,@Ri	Exclusive OR indirect RAM to accumulator	1	2
XRL A,#data	Exclusive OR immediate data to accumulator	2	2
XRL direct,A	Exclusive OR accumulator to direct byte	2	3
XORL direct,#data	Exclusive OR immediate data to direct byte	3	4
CLR A	Clears the accumulator	1	1
CPL A	Complements the accumulator (1=0, 0=1)	1	1
SWAP A	Swaps nibbles within the accumulator	1	1
RL A	Rotates bits in the accumulator left	1	1
RLC A	Rotates bits in the accumulator left through carry	1	1
RR A	Rotates bits in the accumulator right	1	1
RRC A	Rotates bits in the accumulator right through carry	1	1

Bit-oriented Instructions

Similar to logic instructions, bit-oriented instructions perform logic operations. The difference is that these are performed upon single bits.

Mnemonic	Description	Byte	Cycle

CLR C	Clears the carry flag	1	1
CLR bit	Clears the direct bit	2	3
SETB C	Sets the carry flag	1	1
SETB bit	Sets the direct bit	2	3
CPL C	Complements the carry flag	1	1
CPL bit	Complements the direct bit	2	3
ANL C,bit	AND direct bit to the carry flag	2	2
ANL C,/bit	AND complements of direct bit to the carry flag	2	2
ORL C,bit	OR direct bit to the carry flag	2	2
ORL C,/bit	OR complements of direct bit to the carry flag	2	2
MOV C,bit	Moves the direct bit to the carry flag	2	2
MOV bit,C	Moves the carry flag to the direct bit	2	3

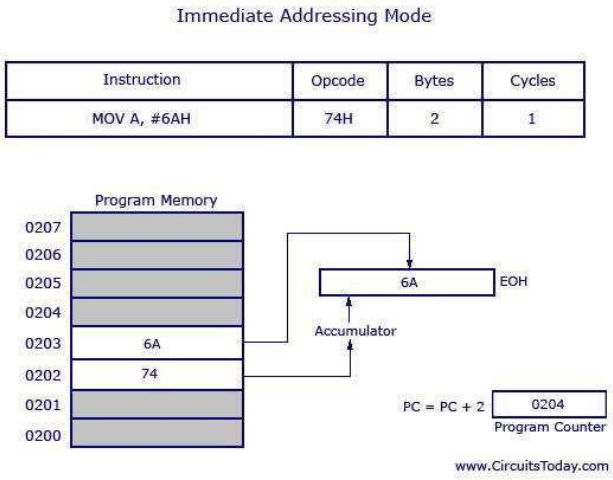
ADDRESSING MODE IN 8051.

Immediate Addressing Mode

MOV A, #6AH

In general we can write MOV A, #data

This addressing mode is named as "immediate" because it transfers an 8-bit data immediately to the accumulator (destination operand).



The picture above describes the above instruction and its execution. The opcode for MOV A, # data is 74H. The opcode is saved in program memory at 0202 address. The data 6AH is saved in program memory 0203..

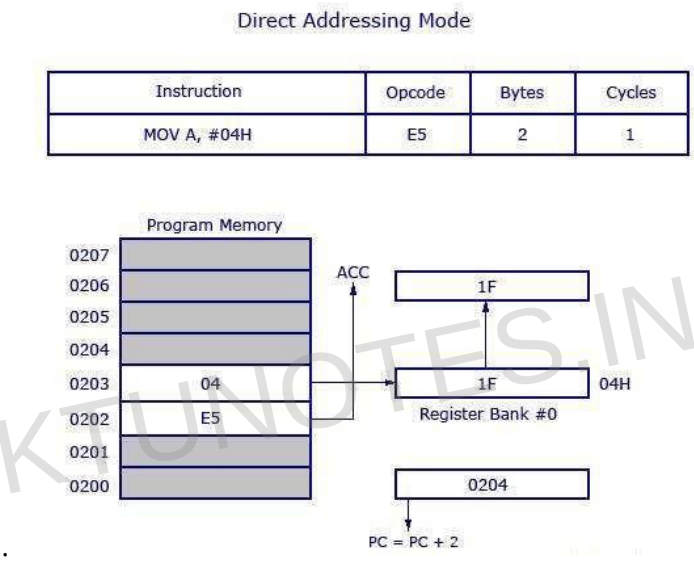
- The *'#'* symbol before 6AH indicates that operand is a data (8 bit). If *'#'* is not present then the hexadecimal number would be taken as address.

Direct Addressing Mode

This is another way of addressing an operand. Here the address of the data (source data) is given as operand. Lets take an example.

MOV A, 04H

Here 04H is the address of register 4 of register bank#0. When this instruction is executed, what ever data is stored in register 04H is moved to accumulator. In the picture below we can see, register 04H holds the data 1FH. So the data 1FH is moved to accumulator.



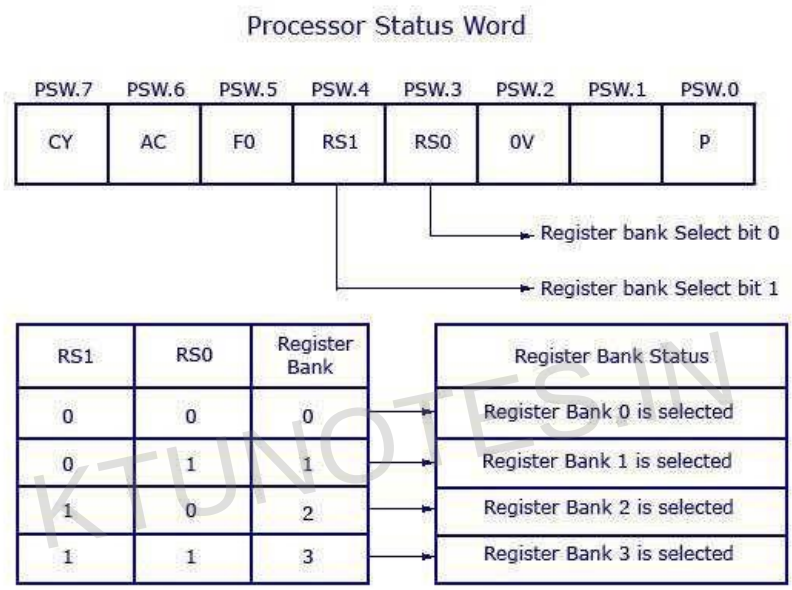
As shown in picture above this is a 2 byte instruction which requires 1 cycle to complete. Program counter will increment by 2 and stand in 0204. The opcode for instruction ***MOV A, address*** is E5H. When the instruction at 0202 is executed (E5H), accumulator is made active and ready to receive data. Then program control goes to next address that is 0203 and look up the address of the location (04H) where the source data (to be transferred to accumulator) is located. At 04H the control finds the data 1F and transfers it to accumulator and hence the execution is completed.

Register Direct Addressing Mode

In this addressing mode the register name directly (as source operand). An example is shown below.

MOV A, R4

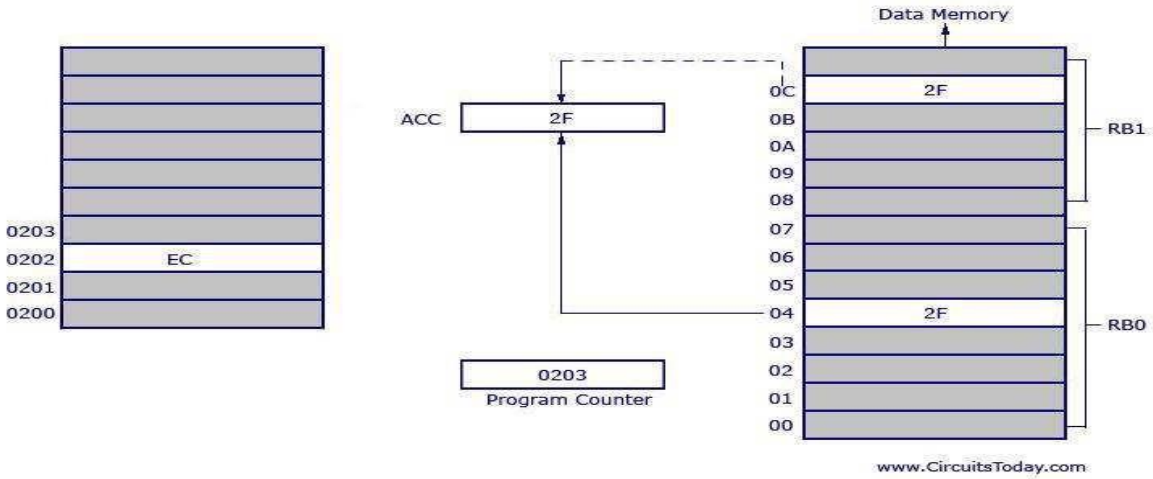
At a time registers can take value from R0,R1…to R7. There are 4 register banks named 0,1,2 and 3. Each bank has 8 registers named from R0 to R7. At a time only one register bank can be selected. Selection of register bank is made possible through a Special Function Register (SFR) named Processor Status Word (PSW). PSW is an 8 bit SFR where each bit can be programmed. Bits are designated from PSW.0 to PSW.7 Register banks are selected using PSW.3 and PSW.4 These two bits are known as register bank select bits as they are used to select register banks. A picture below shows the PSW register and the Register Bank Select bits with status.



So in register direct addressing mode, data is transferred to accumulator from the register (based on which register bank is selected).

Register Direct Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, R4	ECH	1	1



.

Register Indirect Addressing Mode

MOV A, @R0

Register Indirect Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOV A, @ R0	E6H	1	1

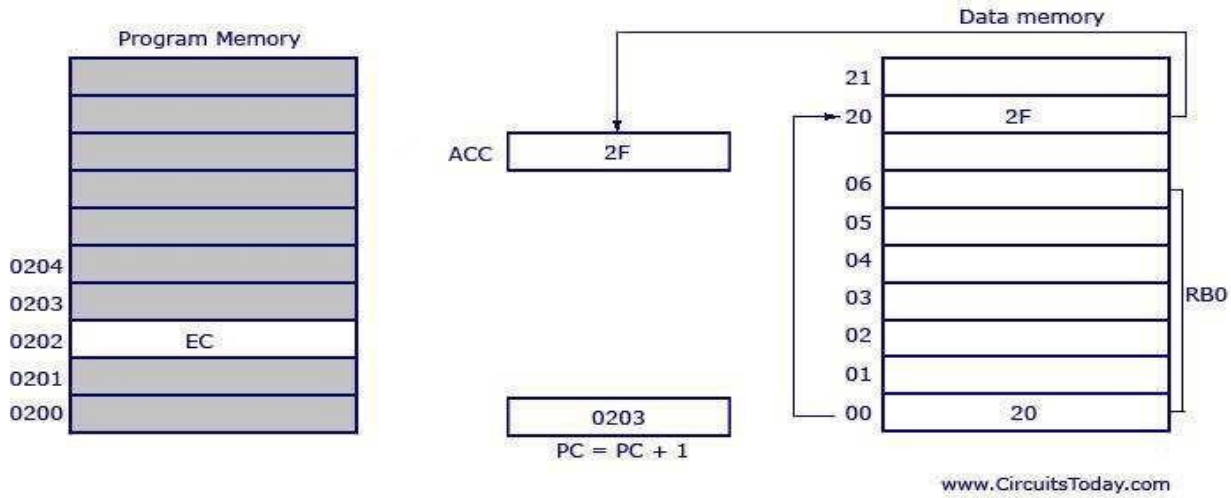
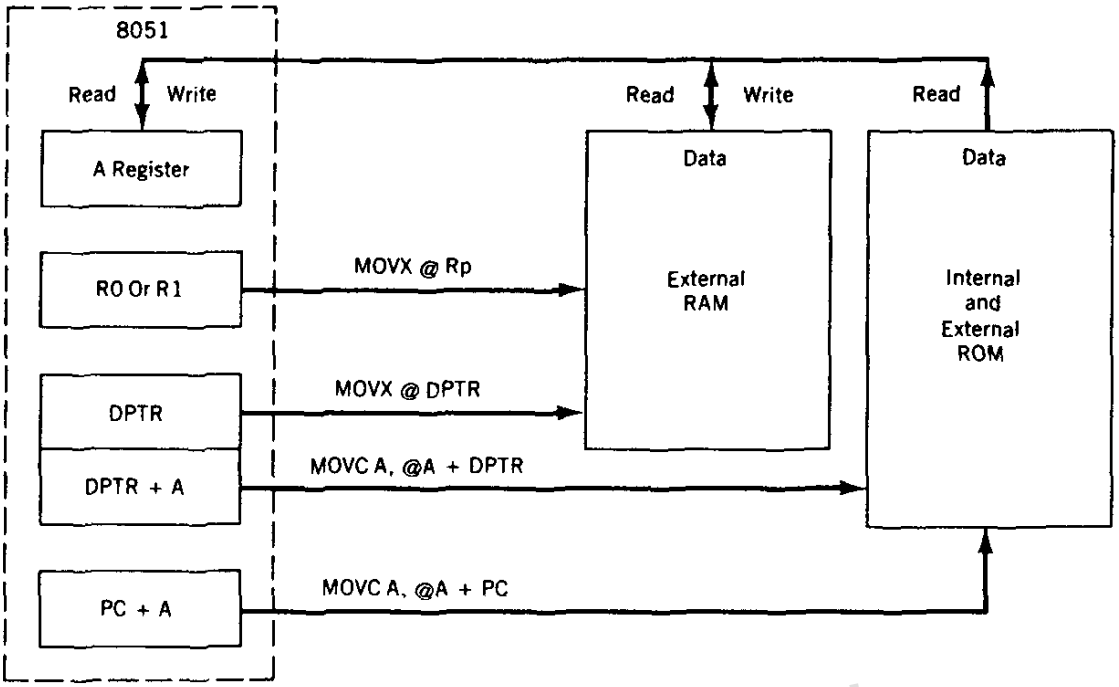


FIGURE 3.2 External Addressing using MOVX and MOVC



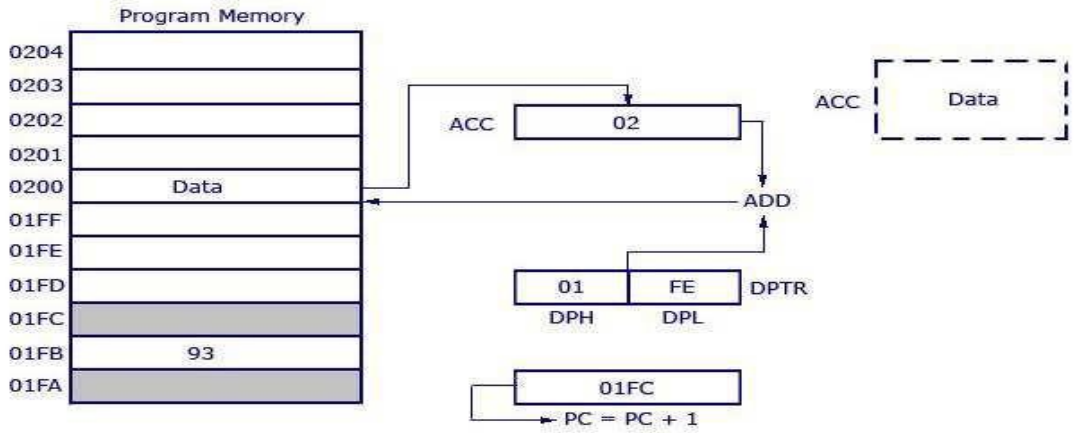
MOVC A, @A+DPTR and MOVC A, @A+PC where DPTR is data pointer and PC is program counter (both are 16 bit registers).

MOVC A, @A+DPTR

The opcode for the instruction is 93H. DPTR holds the value 01FE, where 01 is located in DPH (higher 8 bits) and FE is located in DPL (lower 8 bits). Accumulator now has the value 02H. A 16 bit addition is performed and now 01FE H+02 H results in 0200 H. whatever data is in 0200 H will get transferred to accumulator. The previous value inside accumulator (02H) will get replaced with new data from 0200H. It is a 1 byte instruction. The other example MOVC A, @A+PC works the same way as above example. The only difference is, instead of adding DPTR with accumulator, here data inside program counter (PC) is added with accumulator to obtain the target address.

Indexed Addressing Mode

Instruction	Opcode	Bytes	Cycles
MOVC A,@A +DPTR	93H	1	2



8051 TIMERS AND PROGRAMMING

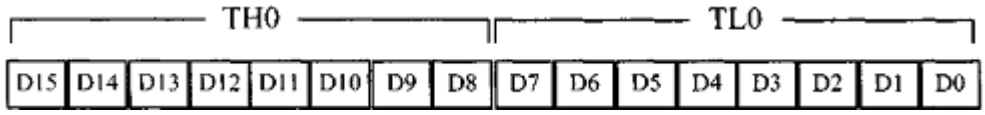
The 8051 has two timers: Timer 0 and Timer 1. They can be used either as timers or as event counters. In this section we first discuss the timers' registers and then show how to program the timers to generate time delays.

Basic registers of the timer

Both Timer 0 and Timer 1 are 16 bits wide. Since the 8051 has an 8-bit architecture, each 16-bit timer is accessed as two separate registers of low byte and high byte. Each timer is discussed separately.

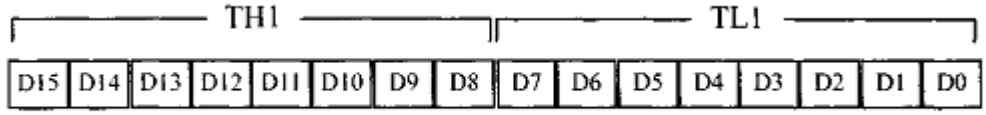
Timer 0 registers

The 16-bit register of Timer 0 is accessed as low byte and high byte. The low byte register is called TLO (Timer 0 low byte) and the high byte register is referred to as THO (Timer 0 high byte). These registers can be accessed like any other register, such as A, B, RO, Rl, R2, etc. For example, the instruction "MOV TLO , #4FH" moves the value 4FH into TLO, the low byte of Timer 0. These registers can also be read like any other register. For example, "MOV R5 , THO" saves THO (high byte of Timer 0) in R5.



Timer 0 Registers

Timer 1 registers

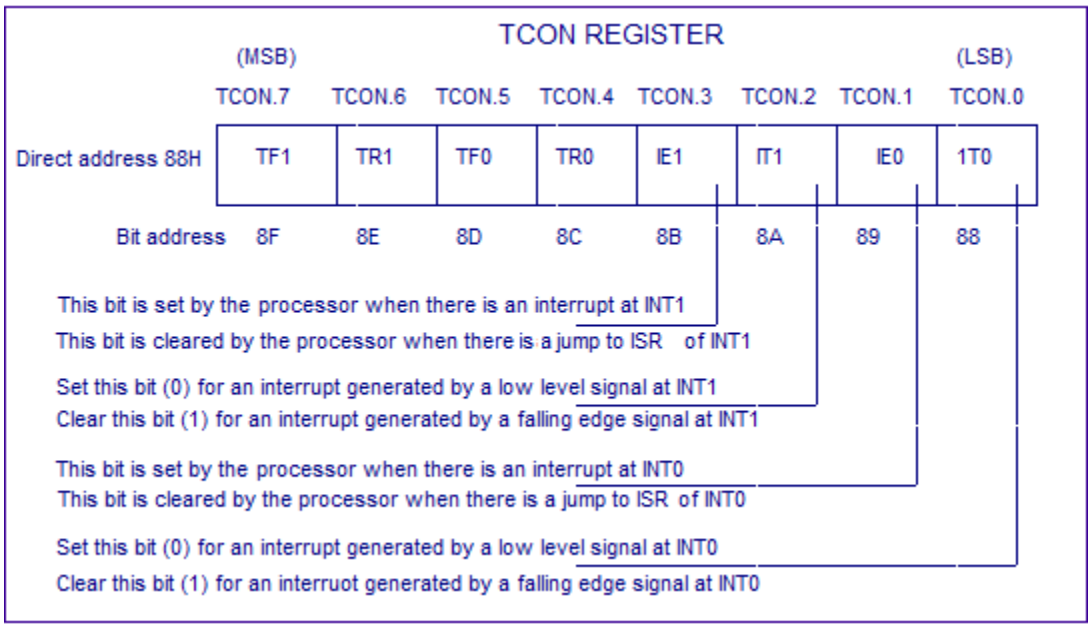


Timer 1 Registers

Timer I is also 16 bits, and its 16-bit register is split into two bytes, referred to as TLl (Timer I low byte) and TH1 (Timer 1 high byte). These registers are accessible in the same way as the registers of timer 0

TCON (Timer/Counter Control Register) control

- TF0/TF1: Timer0/1 overflow flag is set when the timer/counter overflows, reset by program
- TR0/TR1: Timer0/1 run control bit is set to start, reset to stop the timer0/1
- IE0/IE1: **E**xternal **I**nterrupt 0/1 edge detected flag: 1 is set when a falling edge interrupt on the external port 0/1



TMOD (timer mode) register

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 mode and the upper 2 bits to specify the operation.

TMOD Register

(MSB) (LSB)

GATE	C/T	M1	M0	GATE	C/T	M1	M0
Timer 1				Timer 0			

GATE Gating control when set. The timer/counter is enabled only while the INTx pin is high and the TRx control pin is set. When cleared, the timer is enabled whenever the TRx control bit is set.

C/T Timer or counter selected cleared for timer operation (input from internal system clock). Set for counter operation (input from Tx input pin).

M1 Mode bit 1

M0 Mode bit 0

M1	M0	Mode	Operating Mode
0	0	0	13-bit timer mode 8-bit timer/counter THx with TLx as 5-bit prescaler
0	1	1	16-bit timer mode 16-bit timer/counters THx and TLx are cascaded; there is no prescaler
1	0	2	8-bit auto reload 8-bit auto reload timer/counter; THx holds a value that is to be reloaded into TLx each time it overflows.
1	1	3	Split timer mode

M1, MO

MO and Ml select the timer mode. As shown in Figure 9-3, there are three modes: 0, 1, and 2. Mode 0 is a 13-bit timer, mode 1 is a 16-bit timer, and mode 2 is an 8-bit timer. We will concentrate on modes 1 and 2 since they are the ones used most widely. We will soon describe the characteristics of these modes, after describing the rest of the TMOD register.

C/T (clock/timer)

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/T = 0, it is used as a timer for time delay generation. The clock source for the

time delay is the crystal frequency of the 8051. This section is concerned with this choice. The timer's use as an event counter is discussed in the next section.

GATE

Every timer has a means of starting and stopping. Some timers do this by software, some by hardware, and some have both software and hardware controls. The timers in the 8051 have both. The start and stop of the timer are controlled by way of software by the TR (timer start) bits TRO and TR1. This is achieved by the instructions "SETB TR1″ and "CLR TR1″ for Timer 1, and "SETB TRO" and "CLR TRO" for Timer 0. The SETB instruction starts it, and it is stopped by the CLR instruction. These instructions start and stop the timers as long as GATE = 0 in the TMOD register. The hardware way of starting and stopping the timer by an external source is achieved by making GATE = 1 in the TMOD register GATE = 0, meaning that no external hardware is needed to start and stop the timers. In using software to start and stop the timer where GATE = 0. all we need are the instructions "SETB TRx" and "CLR TRx".