Microcontroller and Embedded System (3-1-0)

Module-I (12 Hours)

THE 8051 MICROCONTROLLER: Microcontroller and Embedded Processors, Overview of the 8051Family 8051 ASSEMBLY LANGUAGE PROGRAMMING: Inside the 8051, Introduction to 8051 Assembly Programming, Assembling and Running an 8051 Program, The Program Counter and ROM Space in the8051, Data types and Directives, 8051 Flag Bits and the PSW Register, 8051 Register Banks and Stack.

JUMP, LOOP, AND CALL INSTRUCTIONS: Loop and Jump Instructions, Call Instructions, Time Delay

Generation and Calculation, I/O PORT PROGRAMMING: Pin Description of the 8051, I/O Programming, Bit manipulation

8051 ADDRESSING MODES: Immediate and Register Addressing Modes, Accessing Memory Using Various Addressing Modes

Module-II (10 Hours)

ARITHMETIC INSTRUCTIONS AND PROGRAMS: Unsigned Addition and Subtraction, Unsigned Multiplication and Division, Signed Number Concepts and Arithmetic Operations, LOGIC INSTRUCTIONS AND PROGRAMS: Logic and Compare Instructions, Rotate and Swap Instructions, BCD and ASCII Application Programs.

Module-III (10 Hours)

SINGLE- BIT INSTRUCTIONS AND PROGRAMMING: Single-Bit Instruction Programming, Single-Bit

Operations with CY, Reading Input Pins vs. Port Latch, TIMER/COUNTER PROGRAMMING IN THE

8051: Programming 8051 timers, Counter Programming, 8051 SERIAL COMMUNICATION: Basics of

serial Communication, INTERRUPTS PROGRAMMING: 8051 Interrupts, Programming Timer Interrupts.

Module-IV (8 Hours)

INTERFACING: Interfacing a Stepper Motor, 8051 /31 INTERFACING TO EXTERNAL MEMORY:

Semiconductor Memory, Memory Address Decoding, 8031/51 Interfacing with External ROM, Data Memory

Space, 8031/51 INTERFACING TO THE 8255: Programming the 8255, 8255 Interfacing, and Other Modes of the 8255

Text Book:

1. M.A. Mazdi & J.G. Mazdi; The 8051 Microcontroller and Embedded System, Pearson Education India,

2005.

2. R. Kamal; EMBEDDED SYSTEMS Architecture, Programming and Design; Tata McGraw-Hill Publishing Company Limited; 2003.

THE 8051 MICROCONTROLLER: Microcontroller and Embedded Processors:

The microprocessor is a programmable chip that forms the CPU of a computer. Nowadays, many microprocessor chips are available in the market for users to select from depending on the application.

In general, processor chips can be classified as general-purpose microprocessors, microcontrollers, and DSP processors.

Microcontrollers are processor chips that generally have memory, input ports, and output ports within the chip itself.

Therefore, they can also be called single-chip computers, computer-on-a-chip, or system-on-a-chip.

Microcontrollers are used in machine control applications, where there is no need to change the program.

Equipments that use microcontrollers include computer printers, plotters, fax machines, Xerox machines, telephones, automotive engine control mechanisms, and electronic instruments such as oscilloscopes, multimeters, planimeters, IC testers, etc.

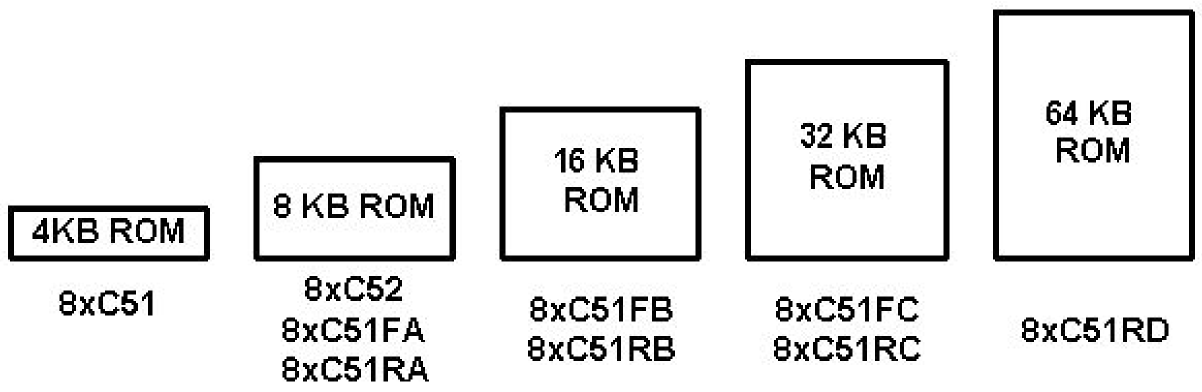
The major difference between microprocessors and microcontrollers is that microcontrollers are comparatively faster because of reduced external memory accessing.

Intel's 8031, 8051, and 8096 and Motorola's 68HC11 are examples of microcontrollers.

Overview of the 8051 Family

Intel Corporation has many micro-controllers in both 8 bit and 16 bit configuration.The 8 bit micro-controllers -in many part numbers -MCS – 51 as the family name.

The various MCS – 51 series micro-controllers. For example, 8XC51RD comes with the internal ROM of 64KB while 8XC51FC comes with only 32KB ROM.



Device Number	**Data bus width**	**RAM capacity**	**ROM capacity**
8031	8	128 bytes	Nil
8051	8	128 bytes	4Kbytes
8751H	8	128 bytes	4Kbytes EPROM
8052AH	8	256 bytes	8Kbytes
8752BH	8	256 bytes	8Kbytes EPROM

Table.1 microcontroller architectures comparision

8051 ASSEMBLY LANGUAGE PROGRAMMING:

i) Introduction to 8051 architecture

With the basic idea on the architecture and the memory organization of 8051 , it is easy to study the instruction set and its flexibility for control applications. Unlike the 8085 instruction set, 8051 instruction set has the instructions for bit manipulations. the 8051 instruction set supports the addressing modes such as indexed addressing and relative addressing

The main features -8051chips are

- 8 bit CPU,
- 4Kbytes of on chip Program memory,

- 128 bytes of on chip data RAM,
- 4 ports of 8bit each,
- Two 16 bit timers,
- Full duplex serial port and
- On-chip clock oscillator.

- In addition to the above features, the 8051 provide Boolean processing; six interrupt capabilities and full-fledged CPU for control applications.

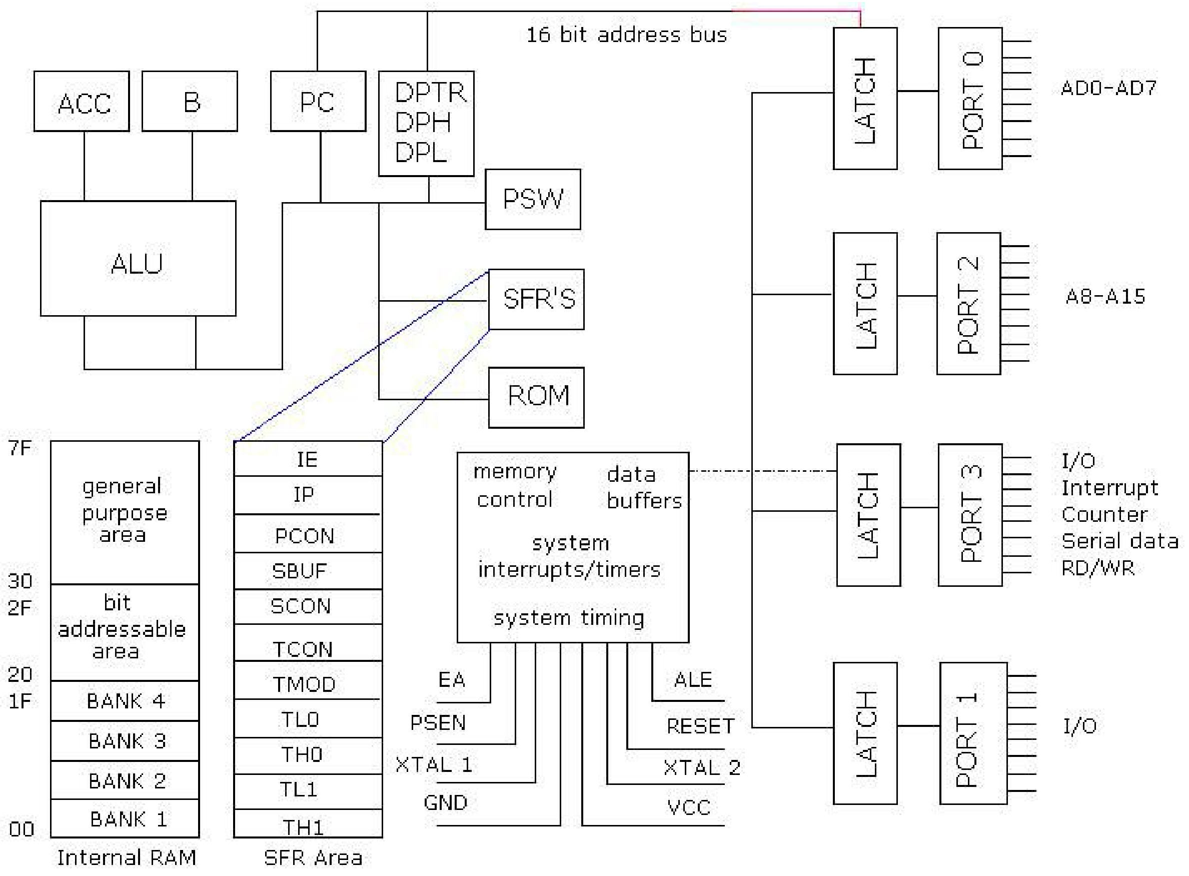


Fig 1. 8051 architecture

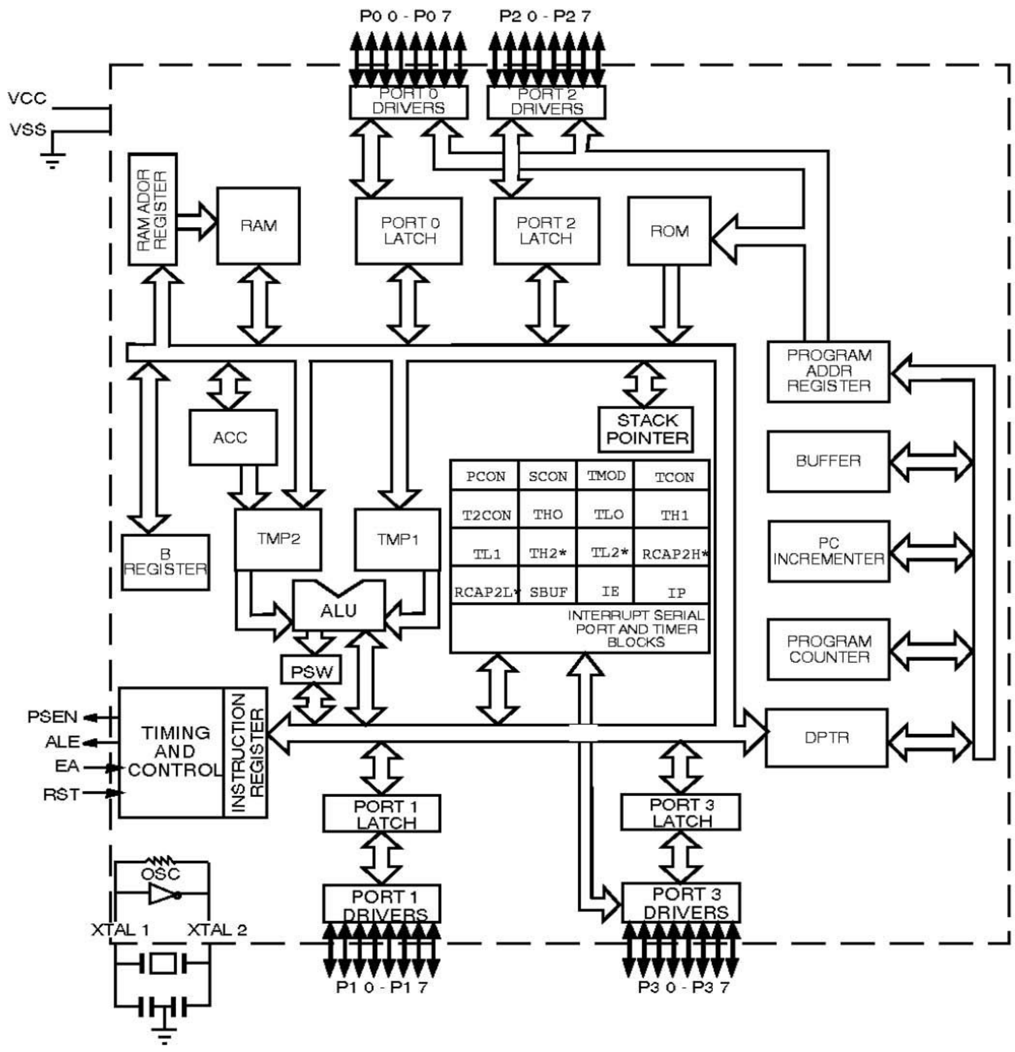


Fig.2. Block diagram of 8051

- 8051 is an 8 bit micro-controller i.e. data bus within and outside the chip is 8 bits wide. The address bus of the 8051 is 16 bits wide, so it can address 64Kbytes of memory. The lower order address bus is multiplexed with the data bus as in -8085 processors. The port0 and port 2 pins of 8051 forms the multiplexed address and data bus.

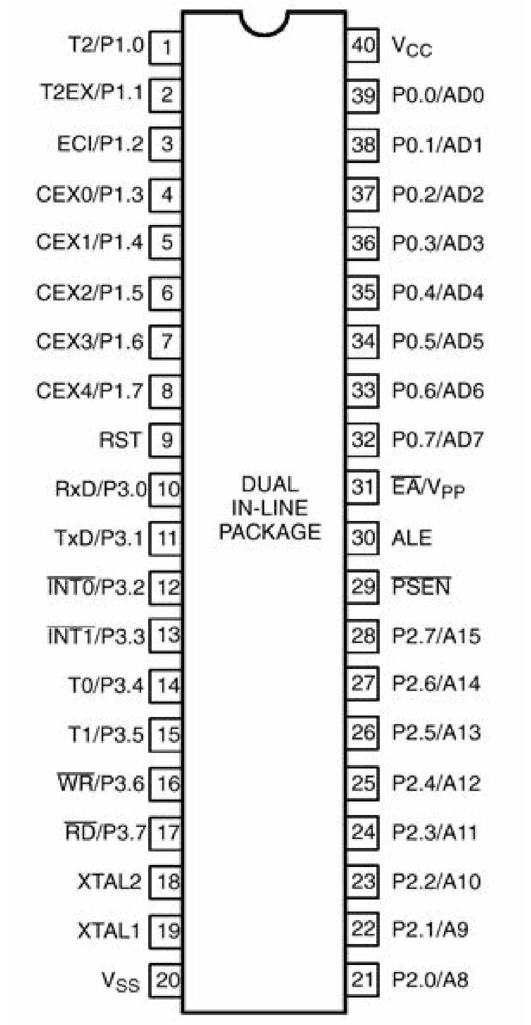


Fig. 3.Pin diagram of 8051

- The 8051 is a 40-pin chip. The power supply +VCC and VSS takes two pins and the built-in clock oscillator requires two pins (–XTAL1 and XTAL2) for connecting the crystal.

  The four control signals pins of 8051 are PSEN, ALE, EA and RST.

  RST is an active high reset signal to restart the controller chip.

  8051 responds to a RST high input only if the RST is held high for at least two machine cycles. A machine cycle is the period taken by any processor to fetch and execute one instruction.

  In 8051, the maximum number of clock cycles taken for a machine cycle is 12. So, the RST pin must be high for at least 24 clock periods.

PSEN, ALE, EA are the signals used in conjunction with the external memory access of the 8051 .

MEMORY ORGANISATION:

In the 8051, the memory is organized logically into program memory and data memory separately. The program memory is read-only type; the data memory is organized as read–write memory.

Again, both program and data memories can be within the chip or outside. The Intel 8051 has 128 bytes of RAM and 4 KB of ROM within the chip.

The address bus of the 8051 is 16 bits wide. So it can access 64 KB of memory.

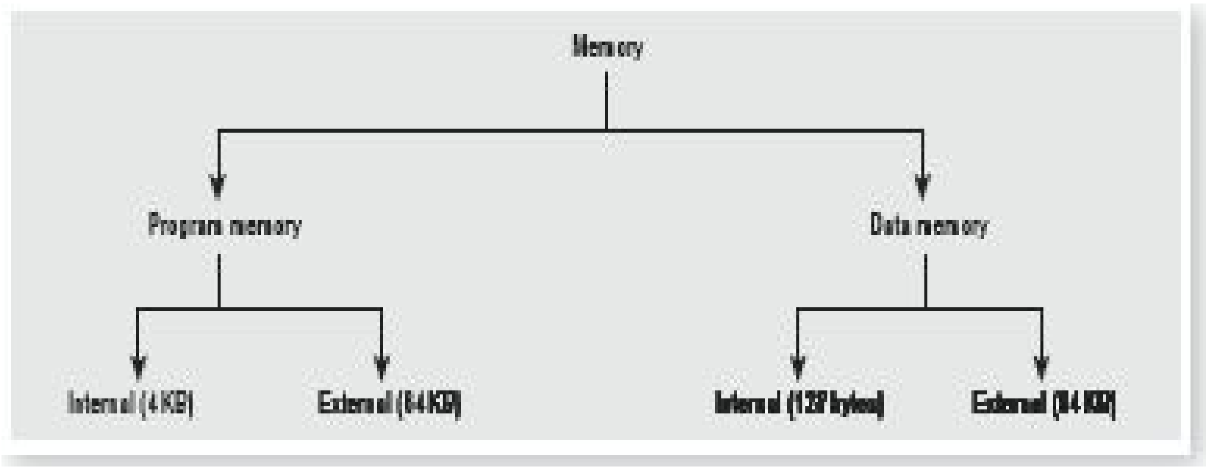


Fig .4 Memory Organization

INTERNAL RAM STRUCTURE:

The 8051 has 128 bytes of internal data RAM, which is accessible as bytes or sometimes as bits.

The address of the internal RAM starts at 00H and occupies space up to 7FH. The RAM space is divided into three blocks—the register banks, the bit-addressable memory, and the scratch pad memory.

The 8051 has four register banks of eight registers each, with addresses from 00H to 1FH. In assembly language, they are addressed by the names R0–R7.

BANK 3	1F	R7
	1E	R6
	1D	R5
	1C	R4
	1B	R3
	1A	R2
	19	R1
	18	R0
BANK2	17	R7
	16	R6
	15	R5
	14	R4
	13	R3
	12	R2
	11	R1
	10	R0
BANK1	0F	R7
	0E	R6
	0D	R5
	0C	R4
	0B	R3

2F	7F	78
2E	77	70
2D	6F	68
2C	67	60
2B	5F	58

7F

	0A	R2	2A	57	50		
	09	R1	29	4F	48		
	08	R0	28	47	40		
	07	R7	27	3F	38		
	06	R6	26	37	30		
	05	R5	25	2F	28		
BANK 0	04	R4	24	27	20		
	03	R3	23	1F	18		
	02	R2	22	17	10		
	01	R1	21	0F	08		
	00	R0	20	07	00	30	

Fig. 5. Internal RAM structure

- The register banks are identified with 2 bits in the processor status word.

  The PSW has two bits for identifying the register bank, i.e., 00 represents bank 0, 01 represents bank 1, 10 represents bank 2, and 11 represents bank 3.

  In the 8051, bitwise operations are also possible with special instructions using the bit addresses. The bit-addressable memory is both bit-addressable (from 00H to 7FH) and byte-addressable (from 20H to 2FH). Bit operations are helpful in many control algorithms.

  Using general-purpose scratch pad memory, programmers can read and write data at any time for any purpose. This memory ranges from the byte address 30H to the address 7FH.

SPECIAL FUNCTION REGISTERS (SFRs):

SFR, which occupies upper 128 bytes of internal memory are the registers, that control the entire processor

They can e accessed by DIRECT addressing.

The registers available in the 8051 are as follows :

- Accumulators - A and B
- Process Status Word - PSW
- I/O port registers - P0, P1, P2, P3
- Data pointers - DPH and DPL
- Serial data buffer register - SBUF
- Stack pointer - SP
- Timer registers - TH0, TH1 and TL0, TL1
- Timer Control Registers - TCON, TMOD
- Power and Port control - PCON, SCON
- Interrupt Control Registers - IP, IE.
- Programmers should not use the addresses in the range 80H to FFH (other than SFR) as it is used by INTEL CORPORATION for expansion functions of 8051.

The 8051 has two accumulators -A register and B register.

The register B forms the accumulator for multiplication and division instructions and for other instructions it can be accessed as a general purpose register.

The stack in the 8051 is organized within the internal RAM area.

The stack pointer is eight bits wide and has to be initialized with an address in the RAM area. When the 8051 is reset, the stack pointer is by default set to 07H. The stack pointer is incremented before storing a data in the stack.

Similarly, while reading data from the stack, the data is read first and then the stack pointer is decremented.

Direct Addressed Memory Address (SFR)	Special Function Register
80	P0
81	SP
82	DPL
83	DPH
87	TCON
88	TMOD
89	TL0
8A	TL1
8B	TH0
8C	TH1
90	P1
98	SCON
99	SBUF
A0	P2
A8	IE
B0	P3
B8	IP

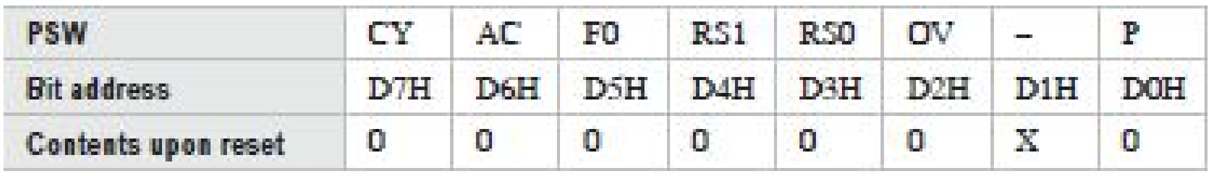
D0	PSW
E0	ACC
F0	B

Fig.6. Special function registers

Processor Status Word:

The PSW contains all the flags of the 8051 and is eight bits wide.The PSW is accessible fully as an 8-bit register, with the address D0H.

The bit pattern of this flag register is



PSW	CY	AC	F0	RS1	RS0	OV	–	P
Bit address	D7H	D6H	D5H	D4H	D3H	D2H	D1H	D0H
Contents upon reset	0	0	0	0	0	0	X	0

Fig.7. Processor status word

- Parity bit (P)

  It is set to 1 if the accumulator contains an odd number of 1s, after an arithmetic or logical operation.

- Overflow flag (OV)

  This flag is set during ALU operations, to indicate overflow in the result. It is set to 1 if there is a carry out of either the D7 bit or the D6 bit of the accumulator. Overflow flag is set when arithmetic operations such as add and subtract result in sign conflict.

- The conditions under which the OV flag is set are as follows:

  - Positive + Positive = Negative
  - Negative + Negative = Positive
  - Positive – Negative = Negative

- Negative – Positive = Positive

Register bank Select Bits(RS1 and RS0)

These bits are user-programmable. They can be set by the programmer to point to the correct register banks.

The register bank selection in the programs can be changed using these two bits.

General-purpose flag (F0)

- This is a user-programmable flag; the user can program and store any bit of his/her choice in this flag, using the bit address.

RS1	**RS0**	**Selected Bank**	Address Range
0	0	Bank 0	00h to 07h
0	1	Bank 1	08h to 0Fh
1	0	Bank 2	10h to 17h
1	1	Bank 3	18h to 1Fh

- Auxiliary carry flag (AC)
  - It is used in association with BCD arithmetic. This flag is set when there is a carry out of the D3 bit of the accumulator.
- Carry flag (CY)
  - This flag is used to indicate the carry generated after arithmetic operations. It can also be used as an accumulator, to store one of the data bits for bit-related Boolean instructions.

The 8051 supports bit manipulation instructions.

This means that in addition to the byte operations, bit operations can also be done using bit data.

For this purpose, the contents of the PSW are bit-addressable.

- 8051 supports bit manipulation instructions. -bit operations can also be done using bit data.
- Similarly, the accumulator and B register contents -bit addressable.
- The bit addresses of all the bits of the accumulator and B registers are given as

POWER DOWN MODE:

- The Power down mode is initiated by making PCON.1 bit to 1.

In this mode, the clock generator -switched off and only the internal memory is active.

the supply voltage Vcc can be reduced to 2V and the power consumption -reduced.

Only way to revoke the processor from power down mode -reset the system.

ADDRESSING MODES OF 8051:

- The way by which a data in specified in an instruction is called as addressing mode.

The data fetched for execution depends upon the addressing mode.

The instruction set of 8051 supports 5 addressing modes

i) Immediate Addressing Mode:
the data to be manipulated is directly given in the instruction itself. The data is preceded by a # symbol.
E.g. ADD A, #80h.

This instruction adds the data 80h to the contents of the accumulator and the result is stored in the accumulator itself. This addressing mode will be used when the data for the arithmetic and logical operation is needed only once and is a constant.

ii) Register Direct Addressing:

The register, that contains the data to be manipulated, is specified in the instruction.

E.g. ADD A, R0.

This instruction will add the contents stored in register R0 with the accumulator contents and store the result in accumulator.

The registers A, DPTR and R0 to R7 are used in Register direct addressing.

This addressing mode uses temporary registers which hold the data for the operation.

iii) Memory Direct Addressing:

The memory address that contains the data to be operated is specified here in the instruction.

E.g. ADD A, 74h.

This instruction adds the data in accumulator with that stored in memory address 74h.

All internal RAM addresses including that of special function registers can be used in memory direct addressing instructions.

This addressing mode is used when the data stored in memory is to be used in arithmetic and logical instructions.

The data in memory used in the direct addressing can be changed at any other point in the program.

iv) Memory Indirect Addressing:

The register, which contains the actual memory address of the data, is specified in the instruction.

The register specified is preceded by @ symbol in assembly language format.

E.g. ADD A, @R0.

The value stored in the register R0 is now the address of the memory location of the data to be fetched from this memory location, the data is fetched and the instruction is executed. The data pointer register (DPTR) is used to access the data in the external memory with 16-bit addresses.

The indirect addressing mode is very much useful for accessing data which are continuously stored in memory and accessed consecutively in program.

v) Indexed Addressing:
In this type of addressing, the instruction consists of two parts - a base address and an offset.
This type of addressing is useful in relative memory accessing and relative jumping.
The base address is stored in data pointer (DPTR) or any other register.
The offset value is stored in Accumulator.
E.g. MOVC A, @A+DPTR.

This instruction adds the contents of the accumulator with the contents of the data pointer and the result forms the actual address of the data from where it is fetched. This data is moved on to the accumulator.

The indexed addressing mode is useful in accessing data structures similar to lookup tables. The base address will hold the address of the starting point of the table and the offset will point the particular entry in the table.

INSTRUCTION SET OF 8051:

- Instruction supported by 8051 can be classified into different types depending upon their operational functions.
- The instruction set classification is as followed.

i) Data Transfer Instructions:

As the name indicates, instructions in this set are used to transfer data.

The data can be transferred from or to external RAM or within the internal memory itself.

The instruction MOV is used to transfer the data between internal registers/memory.

The general format is

a. MOV Reg destination, Reg source.

The source and destination registers within the 8051 chip can be addressed by any one of the addressing modes except indexed addressing mode discussed earlier.

Mnemonic	**Operation**	**Addressing Modes**			
		Direct	**Indirect**	**Register**	**Immediate**
MOV A, <src>	**A = <src>**	**√**	**√**	**√**	**√**
MOV <dest>, A	**<dest> = A**	**√**	**√**	**√**	
MOV <dest>, <src>	**<dest> = <src>**	**√**	**√**	**√**	**√**
MOV DPTR, # data 16	**DPTR = 16-bit immediate data**				**√**
PUSH <src>	**INC SP: MOV “@SP”,**	**√**			

	<scr>				
POP <dest>	MOV <dest>, “@SP”: DEC SP	√			
XCH A, <byte>	ACC and <byte> Exchange Data	√	√	√	
XCHD A, @Ri	ACC and @ Ri exchange low nibbles		√		

MOVX A, @Ri	Copy 8 bit data from the external RAM location pointed to by Ri to register A	Only Indirect Addressing mode
MOVX @ Ri, A	Copy 8-bit data from register A to the external RAM location pointed to by Ri	Only Indirect Addressing mode
MOVX A, @	Copy 8-bit data from the	Only Indirect Addressing mode

DPTR	**external RAM location pointed to by the 16-bit DPTR to register A**	
MOVX @ DPTR, A	**Copy 8-bit data from register A to the external RAM location pointed to by the 16-bit DPTR**	**Only Indirect Addressing mode**
MOVC A, @A + DPTR	**Read Program Memory at (A + DPTR)**	**Only Indirect Addressing mode**
MOVC A, @A + PC	**Read Program Memory at (A + PC)**	**Only Indirect Addressing mode**

Table 2.Data transfer Instructions

The instructions with the mnemonic MOVX is used to access data from external memory locations using indirect addressing only. –

MOVX instruction must use Accumulator (A) register as -destination or source and the other is indirectly accessed external memory.

MOVX can be used -8 bit external memory address and 16 bit external memory address. It can be noted that the external memory -interfaced with 8051 with either 8 bit address or 16 bit address.

If the 8 bit address is used-internal register (any location in Internal RAM) -hold the address of the memory. If 16 bit address is used-Data Pointer (DPTR) is used to hold the address.

The instructions MOVC A,@A+DPTR and MOVC A,@A+PC are the two instructions meaning MOVE CODE MEMORY and are used to transfer data from program memory using indexed addressing.

The program memory addressing using MOVC instruction needs 16 bit address. So, the Data Pointer register (DPTR) and Program Counter (PC) -base registers -instructions.

Data can only be read from the program memory and not written into because the program memory is generally ROM.

PUSH instruction is used to copy data in any internal RAM location to the stack .

The POP instruction is used to copy data from the top of the stack to the RAM location specified in the instruction.

XCHD is used to transfer only the lower-order nibble between the accumulator and the indirectly addressed internal RAM.

XCHD is used to exchange the contents of the accumulator and a register or the internal memory of the 8051.

ii) Arithmetic Instructions:

These instructions are used to do arithmetic operations.
The common arithmetic operations like addition, subtraction, multiplication and division are possible with 8051
All the data used in arithmetic instructions must be available inside the controller i.e. in the internal RAM area only.

ADD instruction is used to add any 8 bit data with Accumulator and the result is stored in Accumulator (A) register. The carry generated if any is stored in Carry flag of the processor status word.
The ADDC instruction is also used to add any 8 bit data with Accumulator along with Carry bit.

- The SUBB instruction -subtract contents of a register from the Accumulator content and during this subtraction, the Carry bit is also subtracted from the accumulator.

  For ADD and SUBB instructions, one of the data must be in Accumulator and the other data - in any direct addressed or indirect addressed internal memory location or can be an immediate data.
- In addition to - ADD, ADDC and SUBB instructions in 8085, -have instructions MUL and DIV.
  The register B is exclusively used for these two instructions. The operands should be stored in the registers A and B for the MUL and DIV instructions.
- The MUL instruction multiplies the contents of A and B registers and stores the 16 bit result in the combined A and B registers.
  The lower order byte -result is stored in A register and the higher order byte - stored in B register.
  The DIV instruction upon execution will divide the contents of A register by the contents of B register.
- The quotient of the result - stored in A register and the remainder is stored in B register.
  A division by 0 i.e. 0 in the B register before executing DIV AB will result in the overflow flag (OV) set to 1.
  DA A instruction -to convert binary sum obtained after adding two BCD numbers into BCD number.

Mnemonic	**Operation**	**Addressing Modes**			
		Direct	**Indirect**	**Register**	**Immediate**
ADD A, <byte>	A = A + <byte>	√	√	√	√
ADDC A,<byte>	A = A + <byte> +C	√	√	√	√

SUBB A, <byte>	A = A – <byte> –C	√	√	√	√
INC A	A = A + 1	Accumulator Only			
DEC A	A = A - 1	Accumulator Only			
DEC <byte>	<byte> = <byte> – 1	√	√	√	√
MUL AB	B:A= B A	Accumulator Only			
DIV AB	A = Int [A/B] B = Mod [A/B]	Accumulator Only			
DA A	Decimal Adjust Accumulator	Accumulator Only			

Table 3.Arithmatic instructions

Logical Instructions:

- In addition to logical AND, OR and XRL operation, 8051 has additional instructions - CLR, CPL. All the data for the logical instructions -available in the internal RAM only.
  The instruction CLR A -to clear the contents of A register, CPL is used to complement or logically invert the contents of the A register and SWAP -to swap the nibbles of A register.

8051 supports four rotate operations with the options –rotating left or right and rotating through carry or not.

Mnemonic	**Operation**	**Addressing Modes**			
		Direct	**Indirect**	**Register**	**Immediate**
ANL A, <byte>	A = A AND <byte>	√	√	√	√
ANL <byte>, A	<byte> = <byte> AND A	√			
ANL <byte>, # data	<byte> = <byte> AND # data	√			
ORL A, <byte>	A = A OR <byte>	√	√	√	√
ORL <byte>, A	<byte> = <byte> OR A	√			
ORL <byte>, # data	<byte> = <byte> OR # data	√			
XRL A, <byte>	A = A XOR <byte>	√	√	√	√
XRL <byte>, A	<byte> = <byte> XOR A	√			
XRL <byte>, # data	<byte> = <byte> XOR # data	√			

CLR A	A = 00H	Accumulator only
CLP A	A =	Accumulator only

	NOT A	
RL A	Rotate ACC Left 1 bit	Accumulator only
RLC A	Rotate Left through Carry	Accumulator only
RR A	Rotate ACC Right 1 bit	Accumulator only
RRC A	Rotate Right through Carry	Accumulator only

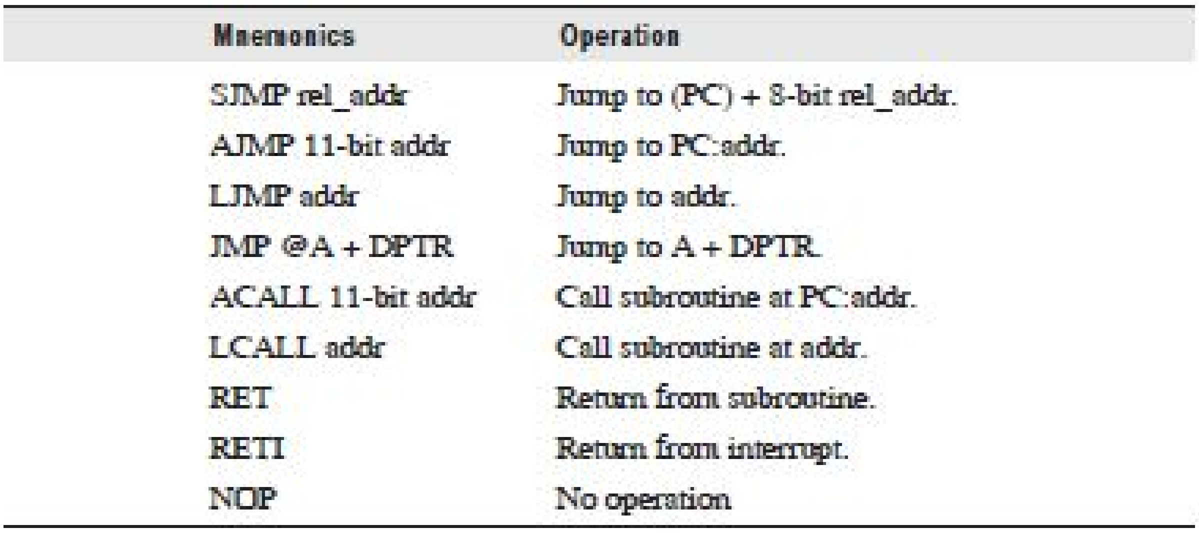
Table 4. logical instructions

Branching Instructions:

- 8051 supports unconditional jumping and subroutine calling in three different ways.

  They are Absolute jump AJMP, ACALL, long jump LJMP, LCALL, and short jump SJMP.

  Un Conditional Branching Instructions



Mnemonics	Operation
SJMP rel_addr	Jump to (PC) + 8-bit rel_addr.
AJMP 11-bit addr	Jump to PC:addr.
LJMP addr	Jump to addr.
JMP @A + DPTR	Jump to A + DPTR.
ACALL 11-bit addr	Call subroutine at PC:addr.
LCALL addr	Call subroutine at addr.
RET	Return from subroutine.
RETI	Return from interrupt.
NOP	No operation

Conditional Branching Instructions

- The syntax for short jump instruction- SJMP 8-bit address.

  This 8 bit address is a relative address- to the program counter.

  The branching address -by adding the address given in the instruction with the program counter content.

  The 8-bit address is a 2's complement number i.e., the most significant bit -sign + or -. The remaining 7 bits - specify the address. using SJMP - branch to anywhere between 127 bytes after the program counter content and 128 bytes before it.(From (PC-128 bytes) to (PC+127 bytes))

  .
- For example,

  8800: SJMP 06h

  This instruction shift the execution to the location 8808h. The program counter content after fetching the 2 byte - SJMP instruction is 8802h. So, 06h added to 8802H results in 8808h.

  The syntax for LJMP -"LJMP 16-bit address". After the execution of this instruction the Program counter -loaded with the 16 bit address and the execution shifts to that location.

  The syntax for AJMP instruction is "AJMP 11 bit jump address".

  The destination branching address -absolute jumping is calculated -keeping MSB 5 bits of the Program counter as it is and changing the LSB 11 bits to that as given -instruction.

  For example,

  8800: AJMP 7F0h

- This instruction branch the execution address 8FF0h. After fetching-program counter content will be 8802h. Keeping the MSB 5 bits of the PC (10001) as it is, and changing the LSB 11 bits to that given in the instruction (111 1111 0000) , the branching address becomes 8FF0h.

  The micro controller 8051 -single instruction for counter operation to decrement -result (DJNZ). -very useful in looping using a counter similar to “for loop” in high level languages.

  Similarly, jumping after checking the result of a comparison -done by a single instruction (CJNE) -very useful for looping of instruction execution based on a condition.

  Used in programming constructs similar to “do while” in high level languages.

  Bit Manipulation Instructions

  The special feature of the 8051 micro controller is that it can handle bit data also like that of byte data.

  The internal data memory map of 8051 has a bit- addressable area also. The special function registers that have the address with 0 or 8 as last digit in their hex address, are also bit addressable.

  The bit manipulation instructions include logical instructions and conditional branching .

Mnemonic	**Operation**
ANL C,bit	C = C AND bit
ANL C,/bit	C = C AND (NOT bit)
ORL C,bit	C = C OR bit
ORL C,/bit	C = C OR (NOT bit)
MOV C,bit	C = bit
MOV bit,C	bit = C
CLR C	C = 0
CLR bit	bit = 0
SETB C	C = 1
SETB bit	bit = 1
CPL C	C = NOT C
CPL bit	bit = NOT bit

JC rel	Jump if C = 1
JNC rel	Jump if C = 0
JB bit,rel	Jump if bit = 1
JNB bit,rel	Jump if bit = 0
JBC bit,rel	Jump if bit = 1 ; CLR bit

Table. 5. Bit Manipulation Instructions

The logical instructions - ANL and ORL. Conditional branching - JC, JNC, JB, JNB, JBC. The other instructions available -CLR, SETB, CPL, and MOV. There are no instructions for halting the machine execution.
Figure shows the flag bits affected by the various instructions.
Increment and decrement instructions do not affect the flag register.
Instructions that affect Flag bits

Instruction	**Flags Affected**		
	C	**OV**	**AC**
ADD	**√**	**√**	**√**
ADDC	**√**	**√**	**√**
SUBB	**√**	**√**	**√**
MUL	**0**	**√**	
DIV	**0**	**√**	
DA	**√**		
RRC	**√**		
RLC	**√**		
SETB C	**1**		
CLR C	**0**		
CPL C	**√**		
ANL C,bit	**√**		
ANL C,/bit	**√**		
ORL C,bit	**√**		
ORL C,/bit	**√**		

MOV C,bit	**√**		
CJNE	**√**		

Table. 6. Instructions that affect Flag bits

Assembler Directives:

The assembler directives are special instruction to the assembler program and are used to define specific operations .

these directives are not part of the executable program.

Some of the most frequently assembler directives are listed as follows:

ORG - OriGinate, defines the starting address for the program in program (code) memory

EQU - EQUate, assigns a numeric value to a symbol identifier so as to make the program more readable.

DB - Define a Byte, puts a byte (8-bit number) number constant at the memory location specified.

DW - Define a Word, puts a word (16-bit number) number constant at the memory location specified.

DBIT - Define a Bit, defines a bit constant, which is stored in the bit addressable section of the Internal RAM.

END - This is the last statement in the source file to advise the assembler to stop the assembly process.

Programming examples using 8051 instruction sets:

- Program to fill a block of memory in internal RAM with a specific data

  Following assembly language program is used to fill the block of internal data memory with a particular DATA. The number of memory locations to be filled is given as COUNT in the program itself. This program uses indirect addressing of the memory location .

- Program:

```
START:      MOV R1, #COUNT          ;load number count
            MOV RO, #30              ;load starting address of memory
                LOOP: MOV @R0, #DATA ;load data to memory location
pointed by R0
            INC R0              ;Point to next memory location
            DJNZ R1, LOOP       ;Check for count and loop
```

Following program uses direct addressing for memory location for achieving the same for the memory locations from 30h to 34h.

- MOV 30, #DATA
- MOV 31, #DATA
- MOV 32, #DATA
- MOV 33, #DATA
- MOV 34, #DATA
- **Example 2**
- Program to add three 8-bit numbers

  The following program is developed assuming that the numbers are in memory locations 30h, 31h and 32h of the internal data RAM and the result is stored in memory locations in 50h and 51h of the internal RAM.
- Algorithm:

  1.The first byte is moved to the accumulator and the second byte is added with it.

  2.If carry flag is set, register R1 is incremented.

  3.The third byte is added with the intermediate result.

  4.If carry flag is set, register R1 is again incremented.

  5.The accumulator forms the least significant byte of the result and register R1 forms the most significant byte of the result.
- Program:

```
START:     MOV R1, #00h      ;Set a register for MSB for result
                 MOV  R0,  #30h       ;Set  starting  address  for  memory
location
            MOV A, @R0          ;Get a data
            INC R0                  ;Point to next memory location
           ADD A, @R0             ;Add the data
           JNC L1                  ;check for carry
              INC R1                   ; If carry is present, increment MSB of
result
        L1:INC R0             ;Point to next memory location
        ADD A, @R0    ;Add the third data
        JNC L2        :Check for carry
        INC R1        ; If carry is p esent, increment MSB of result
   L2:  MOV 50h,A     ; Save the result
        MOV 51h,R1
```

- Example 3
- Program to add two BCD numbers

  The following program is developed assuming that the BCD numbers are in memory locations 30h, and 31h of the internal data RAM and the result is stored in memory locations in 50h and 51h of the internal RAM with the lower order sum in 50h and carry if any in 51h.
- Algorithm:

  The first byte is moved to the accumulator and is added with the second byte.

  The accumulator is now decimal adjusted.

  The value 00h is moved to the accumulator and is added with carry.

  The result is stored in the memory locations 50h and 51h.

```
      MOV A, 30h        ;Get a data
      ADD A, 31h        ;Add the second data
      DAA               ;Decimal adjust accumulator
      MOV 50h,A         ;Save the sum
      MOV A, #00h       ;
      ADDC A, #00h      ;Get the MSB of sum to A register
      MOV 51h,A         ;Save that
```

- Example 4
- Program to add two 16 bit data

In this example, the data are assumed to be initially stored in the external memory locations. First data is stored in locations 4000H and 4001H while the second data is stored in locations 4002H and 4003H.

- Program:

```
      MOV  DPTR, #4000H             ;Point to first data

      MOVX A,@DPTR          ;

      MOV R0,A              ;Get the LSB of first data to R0
```

```
INC DPTR              ;Point to MSB of first data
MOVX A, @DPTR
MOV R1,A              ;Get the MSB of first data to R1
INC DPTR
MOVX A,@DPTR          ;Get the LSB of second data
ADD A, R0             ; Add the LSB of two data
MOV R0,A              ; Store the sum to  the R0 register
INC DPTR
MOVX A,@DPTR          ;Get the MSB of second data
ADDC A,R1             ;Add the MSBs of data along with
 carry out of previous addition
MOV R1,A              ;Store the MSB of sum to R1 register
```

The sum is stored in the R0 and R1 registers at the end of the execution of above program.

8051 Timers :

The 8051 comes -with two 16 bit timers, both of which may be controlled, set, read, and configured individually. The 8051 timers have three general functions:

Programming predefined length of time, and then issuing an interrupt request.

Counting the transitions on an external pin,

Generating baud rates for the serial port.

Basically the timers are the digital counters which are incremented at the pulses given to it. The timers can be controlled to -function through four SFRs namely, TMOD, TCON, TH0/TL0 and TH1/TL1.

The timers will have overflow when it counts to full value and resets to 0 upon next count.

The overflow in the timers will set the two bits in the TCON SFR. This overflow can be programmed to interrupt the microcontroller execution and execute a predefined subroutine .

If the timer registers are incremented by the internal clock pulses from the microcontroller, then the operation is termed as 'Timing' operation.

Meanwhile if the timer registers get their clock pulses from an external device through the port 3 pins of 8051, then the operation is termed as 'Counting'.

Timer 0 external input pin P3.4 (T0) is used give clock input to timer 0 to act as counter.

Timer 1 external input pin P3.5 (T1) is used give clock input to timer 1.

The 8051 has two timers and each of them will have similar operations and functions. The timer in 8051 is basically a 16-bit register which can be incremented depending upon the clock pulses applied to it. These timer registers are configured as the Special Function Registers.

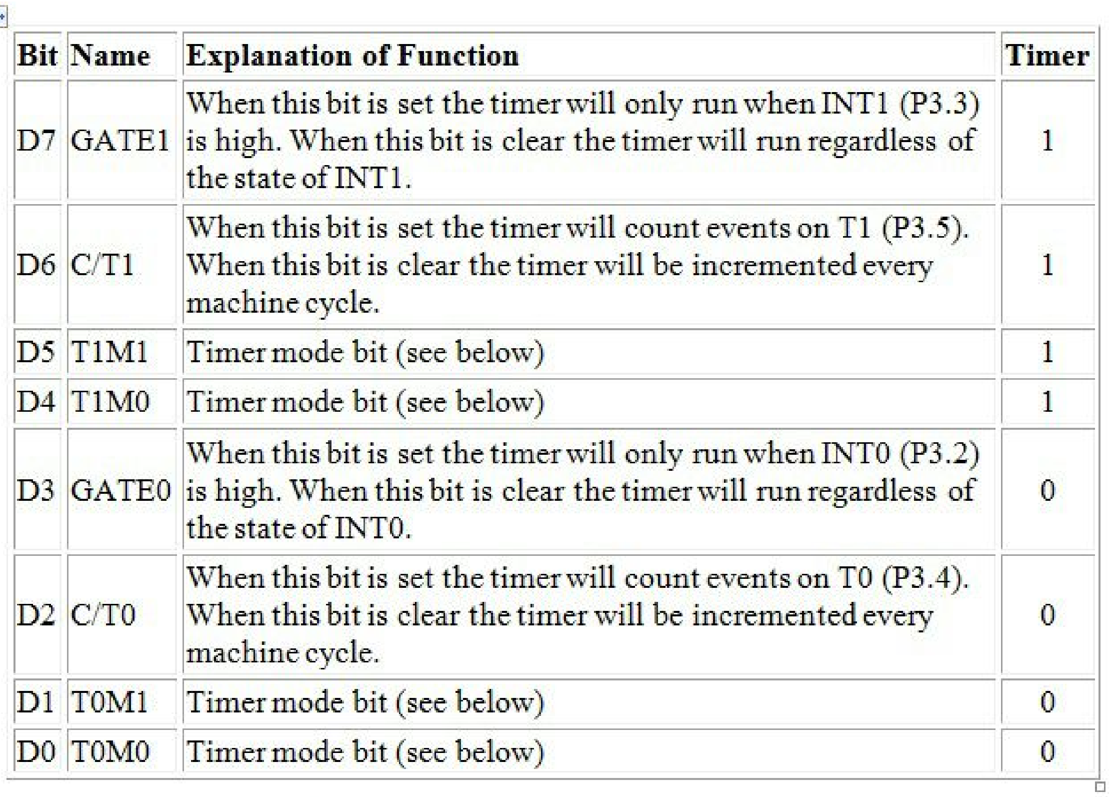
These SFRs at any time has the timer/counter register content. So, the timers can be stopped at any time and the contents can be read from these registers .Since there are only two bytes devoted to the value of each timer it is apparent that the maximum value a timer may have is 65,535.

If a timer contains the value 65,535 and is subsequently incremented, it will reset to 0 with an indication of overflow.

One timer is TIMER0 and the other is TIMER1. Each timer also has two 8 bit SFRs namely TH0 and TL0 forming the higher and lower order bytes of Timer0 and TH1 and TL1 forming the higher and lower bytes of Timer1.

The TMOD SFR -used to control the mode of operation of both timers. Each bit of the SFR gives the micro controller specific information -how to run a timer. The higher order four bits (bits 4 through 7) relate to Timer 1 whereas the low four bits (bits 0 through 3) perform the same functions for timer 0.

BIT Patterns for TMOD SFR :



Bit	Name	Explanation of Function	Timer
D7	GATE1	When this bit is set the timer will only run when INT1 (P3.3) is high. When this bit is clear the timer will run regardless of the state of INT1.	1
D6	C/T1	When this bit is set the timer will count events on T1 (P3.5). When this bit is clear the timer will be incremented every machine cycle.	1
D5	T1M1	Timer mode bit (see below)	1
D4	T1M0	Timer mode bit (see below)	1
D3	GATE0	When this bit is set the timer will only run when INT0 (P3.2) is high. When this bit is clear the timer will run regardless of the state of INT0.	0
D2	C/T0	When this bit is set the timer will count events on T0 (P3.4). When this bit is clear the timer will be incremented every machine cycle.	0
D1	T0M1	Timer mode bit (see below)	0
D0	T0M0	Timer mode bit (see below)	0

Two bits are used for each timer to specify a mode of operation. So, each timer can be operated in any one of four modes.

Table. 7 BIT Patterns for TMOD SFR

SFR that controls the two timers and provides valuable information about them is TCON.

BIT Patterns for TCON SFR

Bit	Name	Bit Address	Explanation of Function	Timer
7	**TF1**	**8Fh**	**Timer 1 Overflow. The micro controller sets this bit when Timer 1**	**1**

			overflows.	
6	TR1	8Eh	Timer 1 Run. When this bit is set Timer 1 is turned on. When this bit is cleared Timer 1 is off.	1
5	TF0	8Dh	Timer 0 Overflow. The micro controller sets this bit when Timer 0 overflows.	0
4	TR0	8Ch	Timer 0 Run. When this bit is set Timer 0 is turned on. When this bit is cleared Timer 0 is off.	0

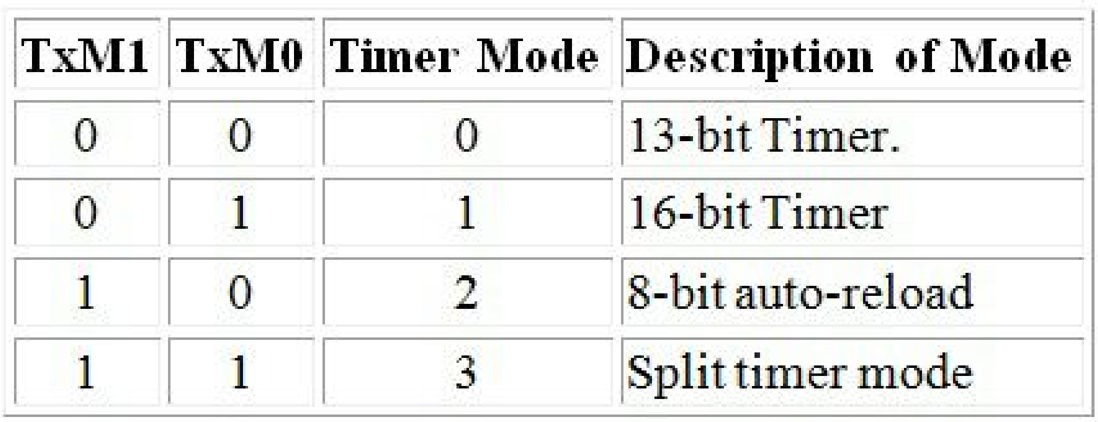
Table. 8 **BIT Patterns for TCON SFR**

TCON SFR:

Only 4 of the 8 bits of the TCON SFR is defined. the other 4 bits of the SFR don’t have anything to do with timers. They are related with Interrupts and they will be discussed in the chapter that addresses interrupts.

Note that the individual bits of TCON register can be addressed separately by their bit addresses. This allows the programmer to run the timers using bit addressable instructions and check the overflow independently.

The two 16 bit timers of 8051 can be operated in any one of the four modes. The mode selection can be done by the setting the proper bits in the TMOD SFR.



TxM1	TxM0	Timer Mode	Description of Mode
0	0	0	13-bit Timer.
0	1	1	16-bit Timer
1	0	2	8-bit auto-reload
1	1	3	Split timer mode

Table. 9 Mode selection in TMOD SFR.

Mode 0 - 13-bit Timer Mode

Timer mode "0" is a 13-bit timer. Out of 16 bits of Timers, only 13bits are used. The 5 bits of lower order byte is used and 8 bits of the higher order byte of the timers are used in Mode0. Lower order byte TL0/1 will count from 0 to 31.

When TL0/1 is incremented from 31, it will "reset" to 0 and increment TH0/1. So, the timer can only contain 8192 values from 0 to 8192.

The timer can be operated as timer with internal clock pulses or as a counter with external clock pulses.

This selection is done by D2 bits of TMOD for Timer 0 and D6 bit of TMOD for Timer1 .The clock pulses selected by D2 and D6 bits of TMOD is then controlled by programmer setting and connected to the Timer registers. The control is by three different means.

First is the Timer Run control bits D4 and D6 of TCON register. The timer will run only when Timer run control bits are set to 1.

The other controls for the timers are through the GATE control bits D4 and D7 of TMOD and the External inputs for timer. Setting GATE to 1 allows the timer to count only if the external control input INT0 or INT1 is set to 1. Setting Gate to 0 will disable the corresponding external timer control inputs INT0 and INT1. Setting Timer to mode 0 will overflow back to zero after 8192 counts. This will set the TF1 and TF0 bits for timer 1 and timer 0 respectively.

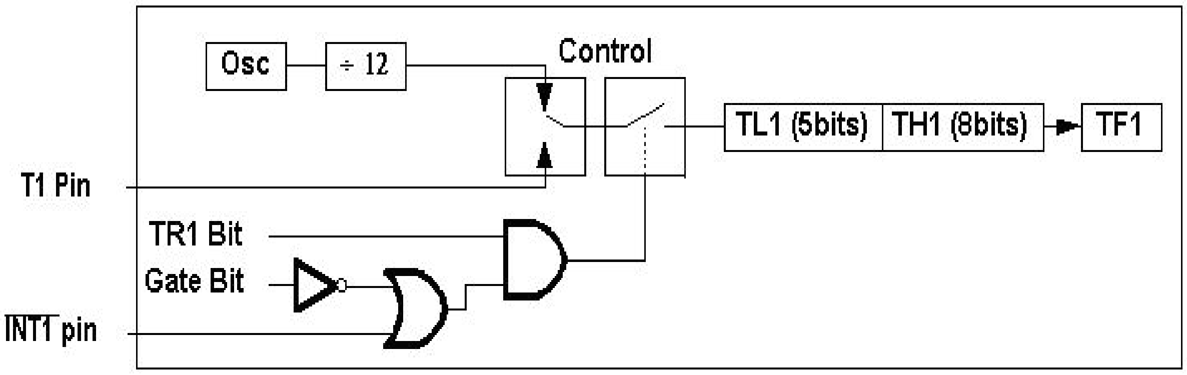


Fig 8. Mode 0 operation

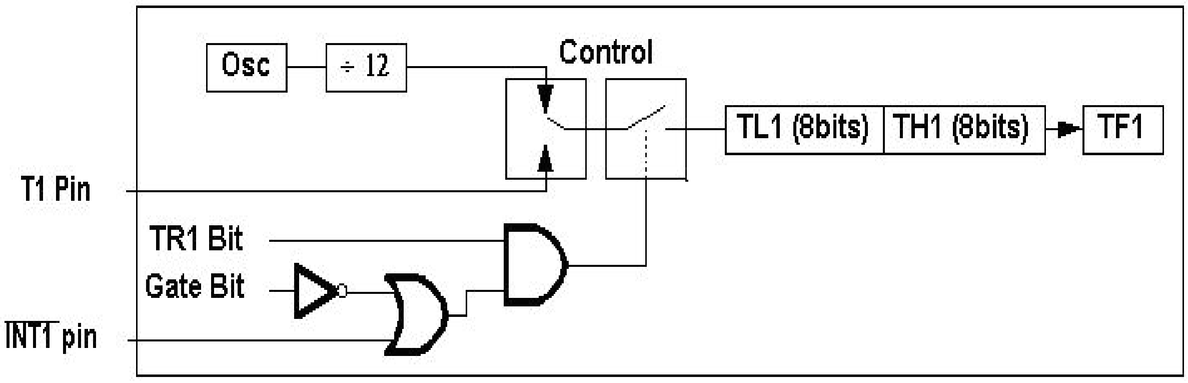


Fig.9. Mode 1 operation

Mode 2 – 8-bit Timer Auto Reload Mode :

Only 4 of the 8 bits of the TCON SFR is defined. the other 4 bits of the SFR don't have anything to do with timers. They are related with Interrupts and they will be discussed in the chapter that addresses interrupts.

Note that the individual bits of TCON register can be addressed separately by their bit addresses. This allows the programmer to run the timers using bit addressable instructions and check the overflow independently.

For example, let's say TH0 holds the value FDh and TL0 holds the value FEh. Then at the next counting pulse, TL0 will be incremented to FFh. Then for the next pulse, the TL0 will overflow and should have become 00.

But, as it is reload mode, the TL0 will be loaded with TH0 i.e., FDh. The value of TH0 will never be changed. TH0/1 is set to a known value and TL0/1 is the SFR that is constantly incremented.

The auto-reload mode is very commonly used for establishing a baud rate for Serial Communications.

The control of gating and running the timer in mode 2 is similar to that of mode 0 .

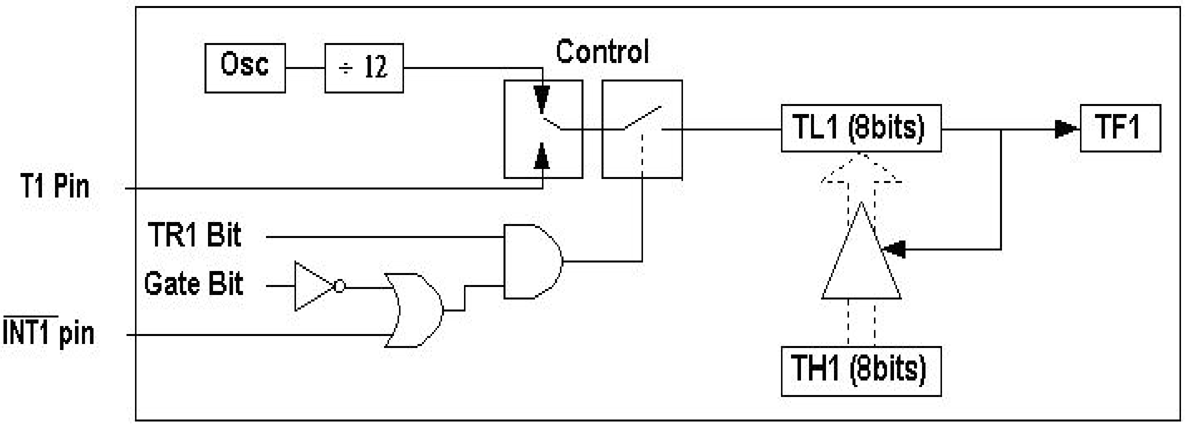


Fig.10. Mode 2 operation

Mode 3 – Split Timer Mode

Mode "3" of 8051 timer is a split-timer mode and is applicable only for Timer 0. When Timer 0 is placed in mode 3, it essentially becomes two separate 8-bit timers. That is, Timer 0 is TL0 and Timer 1 is TH0.

Both timers count from 0 to 255 and overflow back to 0.

In mode 3, all the bits that are related to Real Timer 1 will simply hold its count and will not run and the situation is similar to keeping TR1=0.

In Split Timer mode of Timer 0, the real Timer 1 (i.e. TH1 and TL1) can not be started or stopped since the bits that do that are now linked to TH0. The real timer 1, in this case, will be incremented every machine cycle no matter what.

When two separate timers in addition to a baud rate generator is required in an application, then real Timer 1 can be used as a baud rate generator and TH0/TL0 can be used as two separate timers in mode 3.

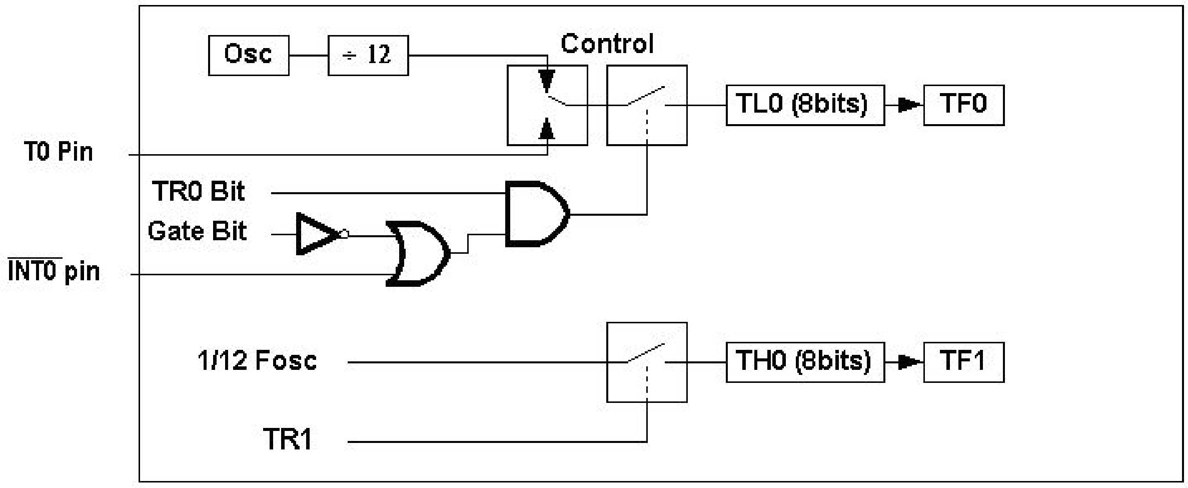


Fig.11 .Mode 3 operation of timer 0 of 8051

Timer Control and Operation

For timer operation (C/T = 0 in TMOD), the timer register counts the divided-down clock. The timer register is incremented once every (FOSC

/ 12) in standard mode. If the clock frequency is 11,059,000KHz, then the counter will be incremented at the rate of (11,059,000KHz/12) = 921,583 KHz.

This means the counter will be incremented 921,583 times in a second. Thus to have delay of say 0.1 seconds, then the counter must be initialized to the count value of (0.1*921,583) = 92158.

Following steps are the program steps to initialize and use a timer in 8051

Decide what mode the timer to be in.

Initialize the TMOD SFR.

Write the timer value to Timer register

Start the timer running by setting the TR0/1 bit in TCON register

Check for TF0/1 bit or program to handle timer overflow as interrupt and execute interrupt subroutine.

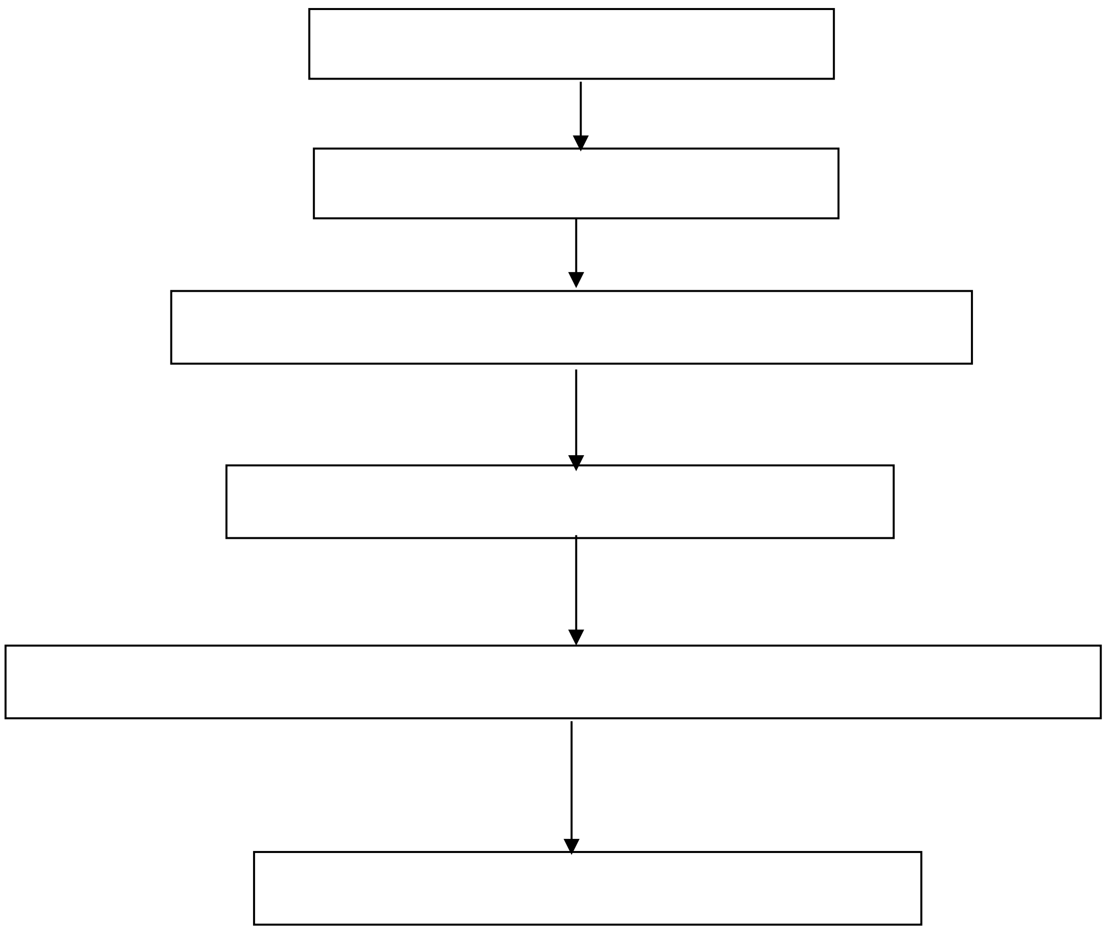


Fig 12. steps in timer control

To set the bit TR1 of TCON (D6 bit), any one of the following two commands can be used -MOV TCON, #40h OR SETB TR1

As, TR1 is a bit addressable location, SETB instruction is used and this has the benefit of setting the TR1 bit of TCON without changing the value of any of the other bits of the SFR.

There are two common ways of reading the value of a 16-bit timer;

1. Read the actual value of the timer as a 16-bit number from TH0/TL0 or TH1/TL1

2. Detect when the timer has overflowed from the TF0/1 bits of TCON. If TF0 is set, it means that timer 0 has overflowed; if TF1 is set, it means that timer 1 has overflowed. This overflow can act as an interrupt and can directly run an Interrupt service routine, if enabled properly.

The timer can be programmed to give an interrupt after a predefined count value. For example, after counting 12 objects in the conveyor or after counting 12 pulses in the timer, an interrupt may be given to the 8051 system to give signal to some other action like packing the 12 items

It is important to note that the 8051 checks the P3.4 line each instruction cycle (12 clock cycles). This means that if P3.4 is low, goes high, and goes back low in 6 clock cycles it will not be detected by the 8051.

This also means the 8051 event counter is only capable of counting events that occur at a maximum of 1/24th the rate of the crystal frequency.

That is to say, if the crystal frequency is 12 MHz, it can count a maximum of 500,000 events per second (12.000 MHz * 1/24 = 500,000).

If the event being counted occurs more than 500,000 times per second, it will not be able accurately counted by the 8051.

Example: In the following program, we create a square wave of 50% duty cycle (with equal portions high and low) on the P1.5 bit. Timer 0 is used to generate the time delay. Analyze the program

```
MOV TMOD,#01 ;Timer 0, mode 1(16-bit mode)

HERE: MOV TL0,#0F2H ;TL0=F2H, the low byte

MOV TH0,#0FFH ;TH0=FFH, the high byte

CPL P1.5 ;toggle P1.5

ACALL DELAY

SJMP HERE
```

In the above program notice the following step.

1. TMOD is loaded.

2. FFF2H is loaded into TH0-TL0.

3. P1.5 is toggled for the high and low portions of the pulse.

8051 interrupts:

8051 basically has following five interrupt sources so that any of the following events will make 8051 to execute an interrupt service routine.

Timer 0 Overflow.

Timer 1 Overflow.

Reception/Transmission of Serial Character.

External hardware interrupt 0.

External hardware interrupt 1.

Different interrupt sources have to be distinguished and 8051 must execute different subroutines depending –interrupt triggered. This is accomplished by jumping or calling to a fixed address when interrupt occurs.

These addresses are called interrupt vector addresses or interrupt handler addresses.

Interrupt	**Flag**	**Interrupt Vect Address**
External 0	IE0	0003h
Timer 0	TF0	000Bh
External 1	IE1	0013h
Timer 1	TF1	001Bh
Serial	RI/TI	0023h

Table. 10 Interrupts in 8051

Whenever Timer 0 overflows (i.e., the TF0 bit is set), the main program will be temporarily suspended and control will jump to 000BH.

It is assumed that service routine at address 0003H handles the situation of Timer 0 overflowing.

Enabling and disabling the interrupts:

By default at power up, all interrupts are disabled. This means that even if, for example, the TF0 bit is set, the 8051 will not execute the interrupt. Programming must be done specifically to enable interrupts.

Interrupt Enable Special Function Register IE SFR at the address A8h is used to enable and disable interrupts by modifying its bits

The interrupts enabling can be handled individually by - bit addresses for the individual bits of IE register.

Bit Patterns for the IE SFR (A8H)

Bit position	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
Bit Address	AF			AC	AB	AA	A9	A8
Name	EA	-	-	ES	ET1	EX1	ET0	EX0
Explanation	Global Interrupt	Undefined	Undefined	Enable Serial	Enable Timer 1	Enable External 1	Enable Timer 0	Enable External 0

	Enable/ Disable			Interrupt	Interrupt	Interrupt	Interrupt	Interrupt

Table. 11 Bit Patterns for the IE SFR (A8H)

Each of the 8051's interrupts has its own bit in the IE SFR. A particular interrupt can be enabled by -corresponding bit. For example, to enable Timer 1 Interrupt, the one of the following instructions can be executed.

MOV IE, #08h OR SETB ET1

However, before Timer 1 Interrupt (or any other interrupt) is truly enabled, bit 7 of IE SFR must also be set. Bit 7, the Global Interrupt Enable/Disable, enables or disables all interrupts simultaneously.

That is, if bit 7 is cleared then no interrupts will occur, even if all the other bits of IE are set. Setting bit 7 will enable all the interrupts -selected by setting other bits in IE.

Interrupt Priorities and Polling Sequence

The 8051 automatically evaluates whether an interrupt occurs after every instruction. When checking for interrupt conditions, it checks them in the following order:

a)External 0 Interrupt

b)Timer 0 Interrupt

c)External 1 Interrupt

d)Timer 1 Interrupt

e)Serial Interrupt

The above list -gives the interrupt priority.

So, whenever the External 0 interrupt and Timer 1 interrupt occurs at the same instant, then 8051 microcontroller executes the interrupt service routine corresponding to External 0 interrupt first.

Then 8051 microcontroller will return to the main program, execute one instruction and then execute the interrupt service routine corresponding to Timer 1 Interrupt.

If a Serial Interrupt occurs at the exact same instant that an External 0 Interrupt occurs, the External 0 Interrupt will be executed first and the Serial Interrupt will be executed once the External 0 Interrupt has completed.

The 8051 offers two levels of interrupt priority: high and low. By using interrupt priorities, the above interrupts can be divided into two separate interrupt priorities. So, the five interrupts can be again prioritized.

Interrupt priorities are controlled by the IP SFR (B8h). For example, if the Serial Interrupt is much more important than the Timer 0 interrupt, then the Interrupt Priority register IP SFR at the address B8h can be properly programmed to set the priority.

This is done by assigning a high priority to the Serial Interrupt and a low priority to the Timer 0 Interrupt. By setting the D4 bit to 1, the serial interrupt will be set to higher priority and making D1 bit to 0, the Timer 0 interrupt will be set to lower priority.

Note that the priority can be set individually by using the bit addresses of the IP register. For example, the timer 0 interrupt priority can be made high by setting the D1 bit of IP SFR. So, the following instructions can be used for the same.

SETB PT0 (or) SETB B9H (or) MOV IP, #82H

Bit Patterns for the IP SFR

Bit position	D7	D6	D5	D4	D3	D2	D1	D0
Bit Address				BC	BB	BA	B9	B8
Name	EA	-	-	PS	PT1	PX1	PT0	PX0
Explanation	Enable Interrupts - Made 0 to disable all interrupts	Undefined	Undefined	Serial Interrupt Priority	Timer 1 Interrupt Priority	External 1 Interrupt Priority	Timer 0 Interrupt Priority	External 0 Interrupt Priority

Table. 12 Bit Patterns for the IP SFR

When considering interrupt priorities, the following rules apply:

Nothing can interrupt a high-priority interrupt--not even another high priority interrupt.

A high-priority interrupt may interrupt a low-priority interrupt.

A low-priority interrupt may only occur if no other interrupt is already executing.

If two interrupts occur at the same time, the interrupt with higher priority will execute first. If both interrupts are of the same priority the interrupt which is serviced first by polling sequence will be executed first.

The five interrupt sources are passed first - IE register, which decides the enabling and disabling of interrupts.

The IP register - set two priority levels among the available interrupts. This is shown in the figure as high priority and low priority blocks. The bits IT0 and IT1 can be set by TCON special function register and this is used to select whether the hardware interrupt is level triggered or edge triggered.

Programming timer interrupts:

Show the instructions to (a) enable the serial interrupt, timer 0 interrupt, and external hardware interrupt 1 (EX1),and (b) disable (mask) the timer 0 interrupt, then (c) show how to disable all the interrupts with a single instruction.

Solution:

```
(a) MOV IE,#10010110B ;enable serial,
;timer 0, EX1
Another way to perform the same manipulation is
SETB IE.7 ;EA=1, global enable
SETB IE.4 ;enable serial interrupt
SETB IE.1 ;enable Timer 0 interrupt
SETB IE.2 ;enable EX1
(b) CLR IE.1 ;mask (disable) timer 0
;interrupt only
(c) CLR IE.7 ;disable all interrupts
```

Basics of serial communication

Computers transfer data in two ways:

i) Parallel:Often 8 or more lines (wire conductors) are

used to transfer data to a device that is only a few feet away.

ii)Serial:To transfer to a device located many meters

away, the serial method is used. The data is sent one bit at a time.

At the transmitting end, the byte of data must be converted to serial bits using parallel-in-serial-out shift register At the receiving end, there is a serialin-parallel-out shift register to receive the serial data and pack them into byte

When the distance is short, the digital signal can be transferred as it is on a simple wire and requires no modulation If data is to be transferred on the telephone line, it must be converted from 0s and 1s to audio tones.
Serial data communication uses two methods
i)Synchronous method transfers a block of
data at a time
ii)Asynchronous method transfers a single
byte at a time
It is possible to write software to use either of these methods, but the programs can be tedious and long. There are special IC chips made by many manufacturers for serial communications
UART (universal asynchronous Receivertransmitter)
USART (universal synchronous-asynchronous Receiver-transmitter)
If data can be transmitted and received,it is a duplex transmission. If data transmitted one way a time, it is referred to as half duplex If data can go both ways at a time, it is full duplex This is contrast to simplex transmission.

A protocol is a set of rules agreed by both the sender and receiver on

i)How the data is packed

ii)How many bits constitute a character

iii)When the data begins and ends

An interfacing standard RS232 was set by the Electronics Industries Association

(EIA) in 1960 The standard was set long before the advent of the TTL logic family, its input and output voltage levels are not TTL compatible.In RS232, a 1 is represented by -3 ~ -25 V,while a 0 bit is +3 ~ +25 V, making -3 to +3 undefined.

Interfacing Stepper motor

Stepper motors are used for position control applications, such as for the control of the disk drives and in robotics. The most common stepper motors have four stator windings that are paired with a center tapped common shown in below Figure .

While the conventional motor shaft runs freely, the stepper motor shaft moves in a fixed repeatable increment, which allows one to move it to a precise position.

The typical stepper motor considered here has 50 teeth on the rotor and 8 poles on the stator for a 1.8º step angle**.**

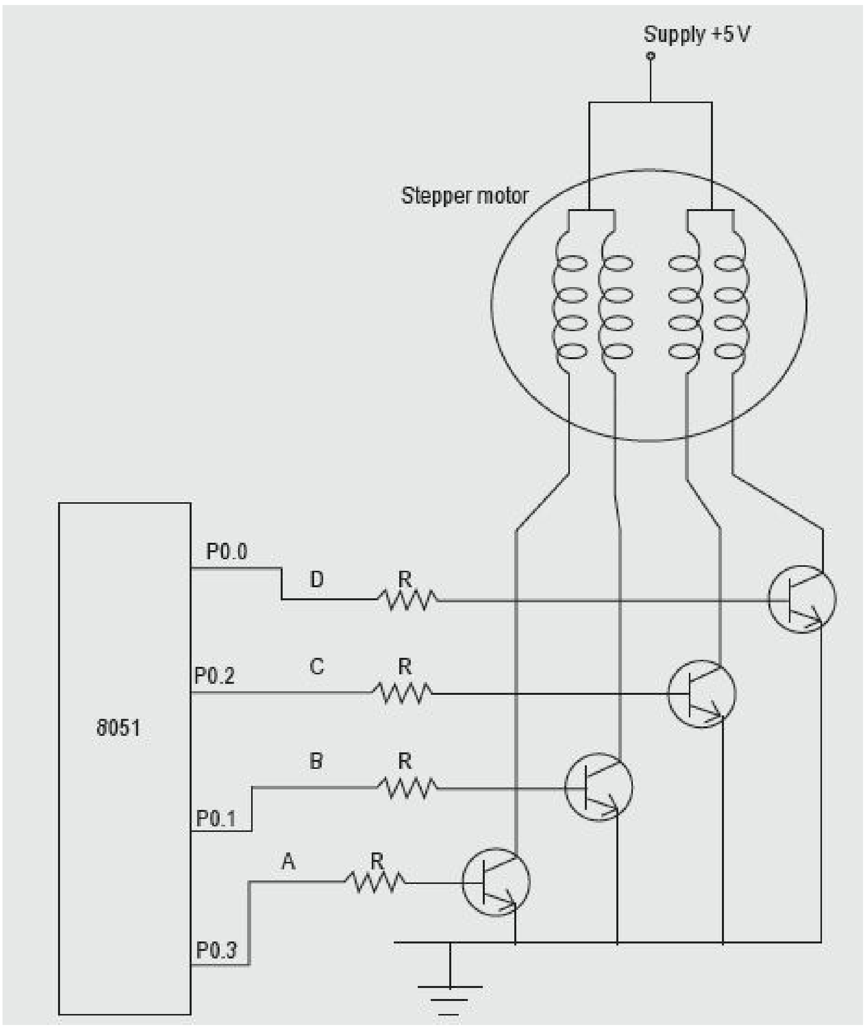


Fig.13 connection of stepper motor to 8051

Interfacing 8255 to 8051

When the 8051 is connected to external memory, port 0 (P0) is used for the lower-order address and data bus and port 2 (P2) is used for the higher-order address bus.

Since the port 3 pins have an alternative function, the net result is that only P1 is left for input and output operation.

One way to expand the number of I/O ports is to connect the 8255 programmable peripheral interface with the 8051.

The interfacing of the 8255 with the 8051 is done assuming the 8255 as a memory location, because the 8051 supports only memory-mapped I/O.

For accessing the external memory in the 8051, the MOVX instruction is used.

The lower-order address bus and the data bus are multiplexed and are available in the port 0 pins.

This is de-multiplexed using a latch and the ALE signal.

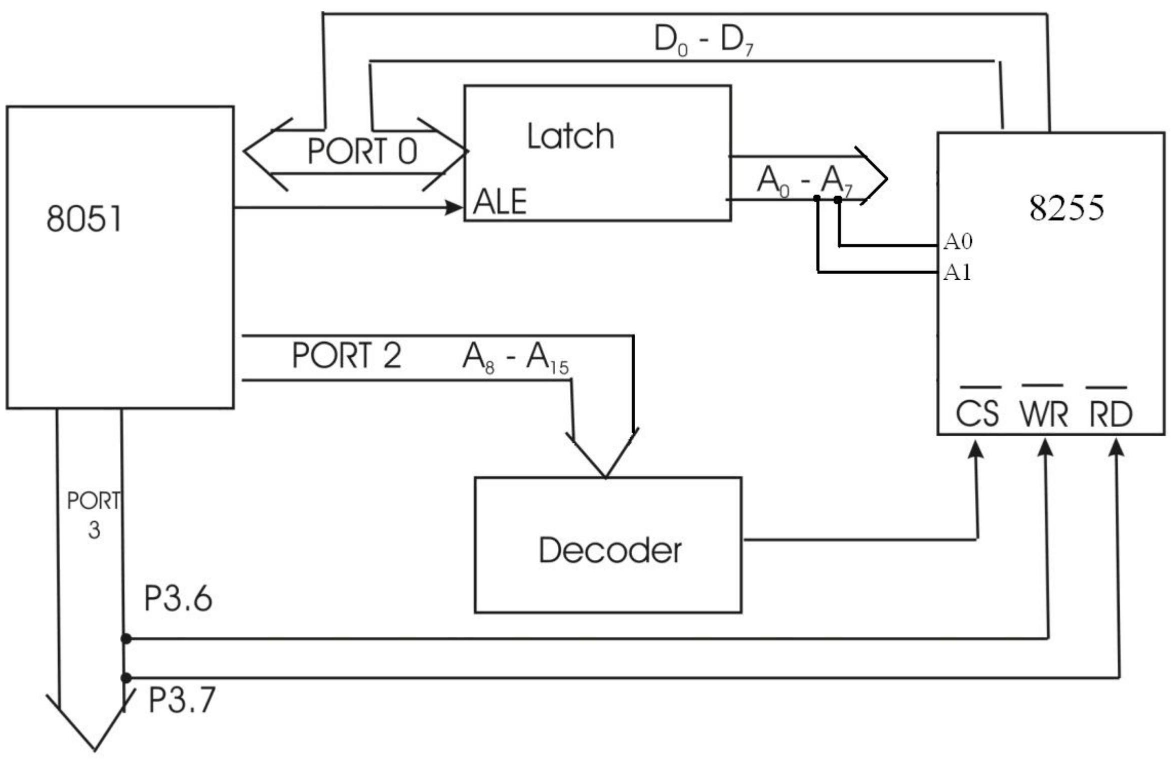


Fig.14 interfacing 8255 to 8051

The first step in the general interfacing method is to decide the addresses for the port.

The 8051 uses 16-bit addresses and the most significant address lines are used for decoding and selecting the device.

Here, the higher-order address bus from port 2 is given to a decoder logic circuit.

From the decoder, the 8255 chip select signal is generated.

The 8255 needs four addresses for interfacing with any processor—three for the ports A, B, and C and one for the control register.

The lower-order address lines A0 and A1 are connected to select one of these four registers. The read and write control signals are available from the port 3 pins P3.7 and P3.6.

Example: Program PC4 of the 8255 to generate a pulse of 50 ms with 50% duty cycle.

Solution:

To program the 8255 in BSR mode, bit D7 of the control word must be low. For PC4 to be high, we need a control word of “0xxx1001”. Likewise, for low we would need “0xxx1000” as the control word. The x’s are for “don’t care” and generally are set to zero.

```
MOV a,#00001001B ;control byte for PC4=1
MOV R1,#CNTPORT ;load control reg port
MOVX @R1,A ;make PC4=1
ACALL DELAY ; time delay for high pulse
MOV A, 00001000B ; control byte for PC4=0
MOVX @R1, A ;make PC4=0
ACALL DELAY
```

External Memory Interfacing in 8051

External memory interfaced to 8051 can be of two types -external program memory and external data memory.

The external memory accesses are accomplished with the Ports 0 and 2 of 8051 as they serve as the multiplexed address/ data buses. The external memory in 8051 is always accessed with 16 bit addresses.

The 8051 outputs the signal ALE (Address Latch enable) in order to demultiplex the lower order address and data bus.

In addition, the micro-controller sends the control signals on the Port3 lines.

i)Program memory interfacing

The program memory can be placed outside the chip in addition to the internal program memory. The complete program memory can be placed outside the chip neglecting the internal program memory.

Applying proper the voltage level on the input line EA of 8051 can do the selection of any of the above two methods.

Connecting EA to Ground will disable the internal program memory and all program memory accesses are done to external memory. The Read strobe signal given by the micro-controller is PSEN.

This active low signal is connected to the Read selection line of the memory chips.

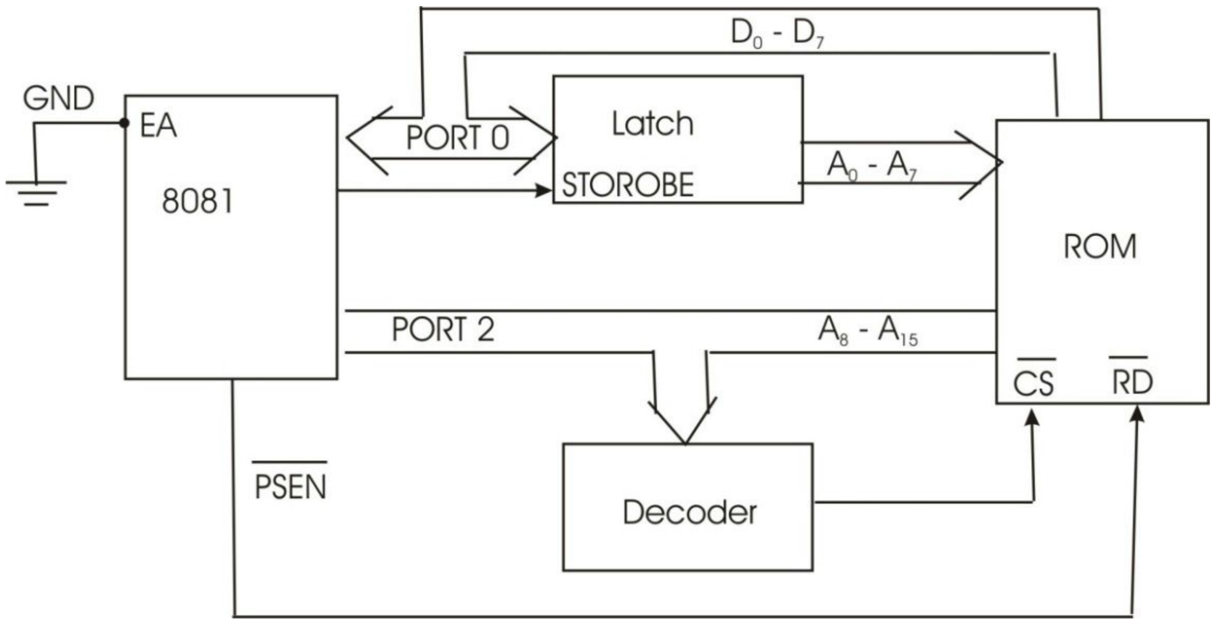


Fig.15connecting external program memory to 8051

Connecting EA pin of 8051 to the logic 1 or +5V will program the microcontroller to use the internal program memory for the addresses starting from 0000. After the available internal memory, the external memory is accessed.

If = 0; The internal program memory is not accessed.

If = 1; Then internal program memory is accessed for address 0000-0FFF (or the available range) and external program memory is accessed for addresses greater than 0FFF.

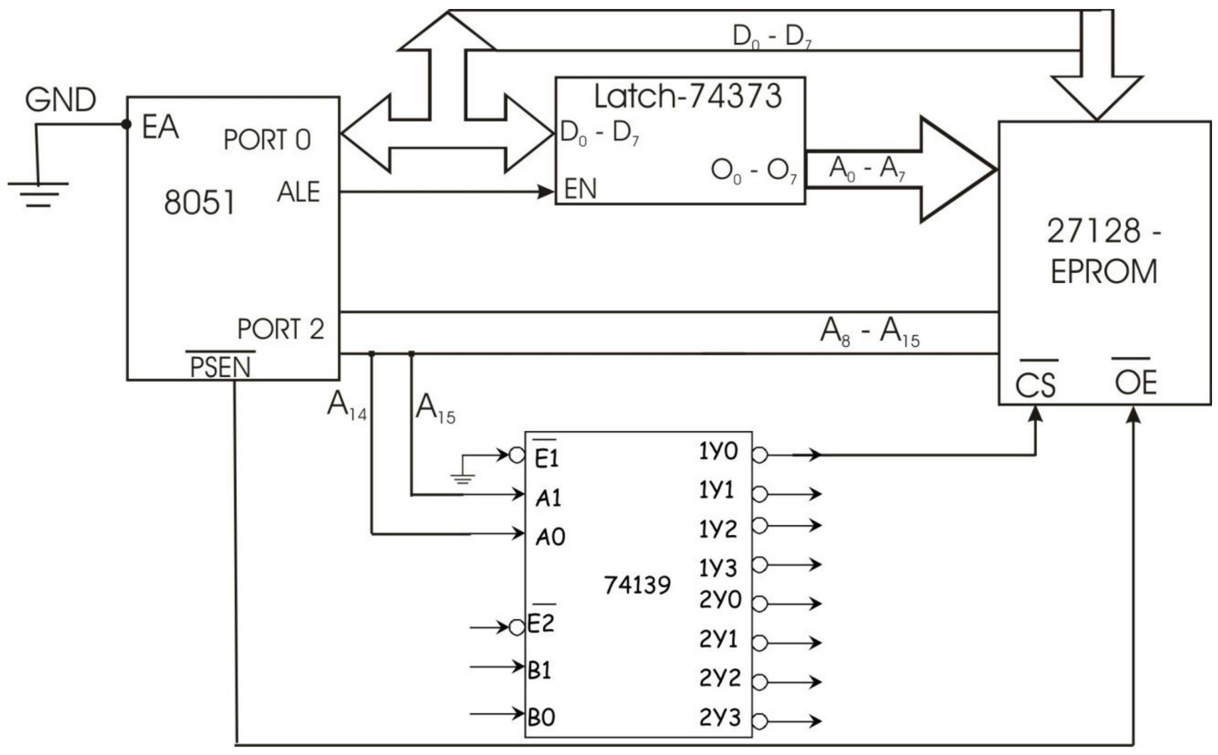


Fig. 16 Connecting EPROM IC 27128 to 8051

iii) Accessing External Data Memory

The data memory in the system - be Random Access memory as it should facilitate both read and write operation of data. The external data memory is interfaced in the same way as the program memory is interfaced.

The major difference is that the read and writes operations can be done in Read /Write Random Access Memory.

The control signals for reading and writing to data memory are available from the port3 pins. P3.6 pin gives data memory write enable signal and P3.7 pin gives out the RAM read enable signal.

A decoder logic circuit is necessary to select the RAM chip based on the higher order address lines.

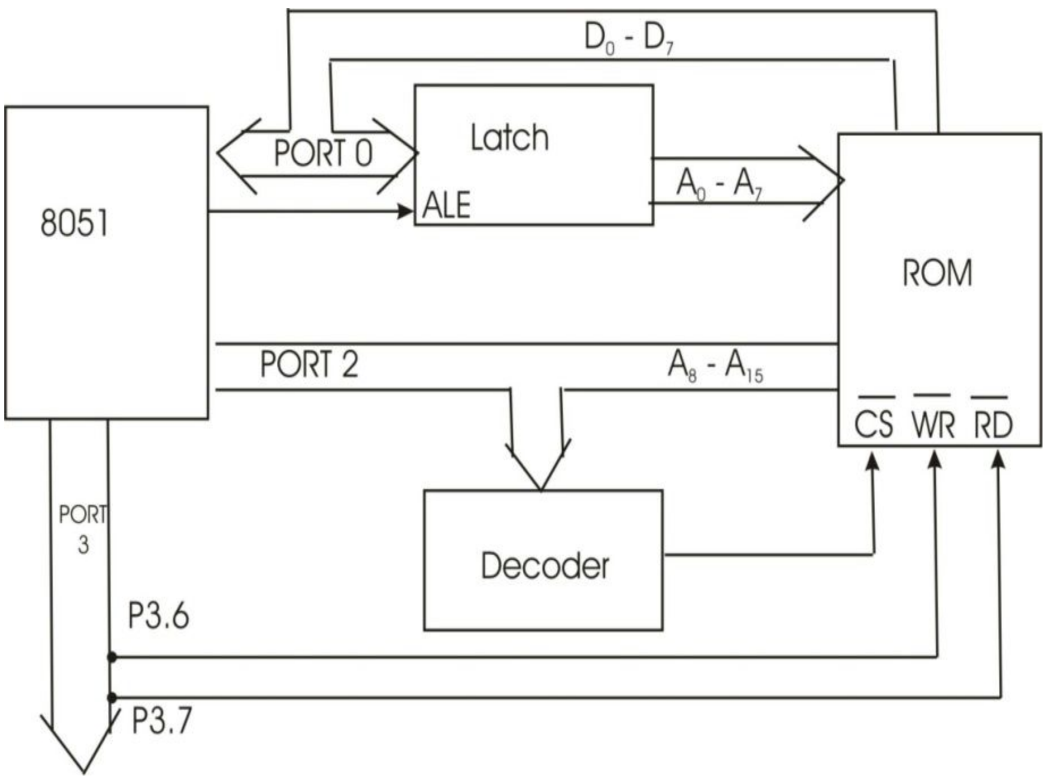


Fig. 17 connecting external data memory to 8051

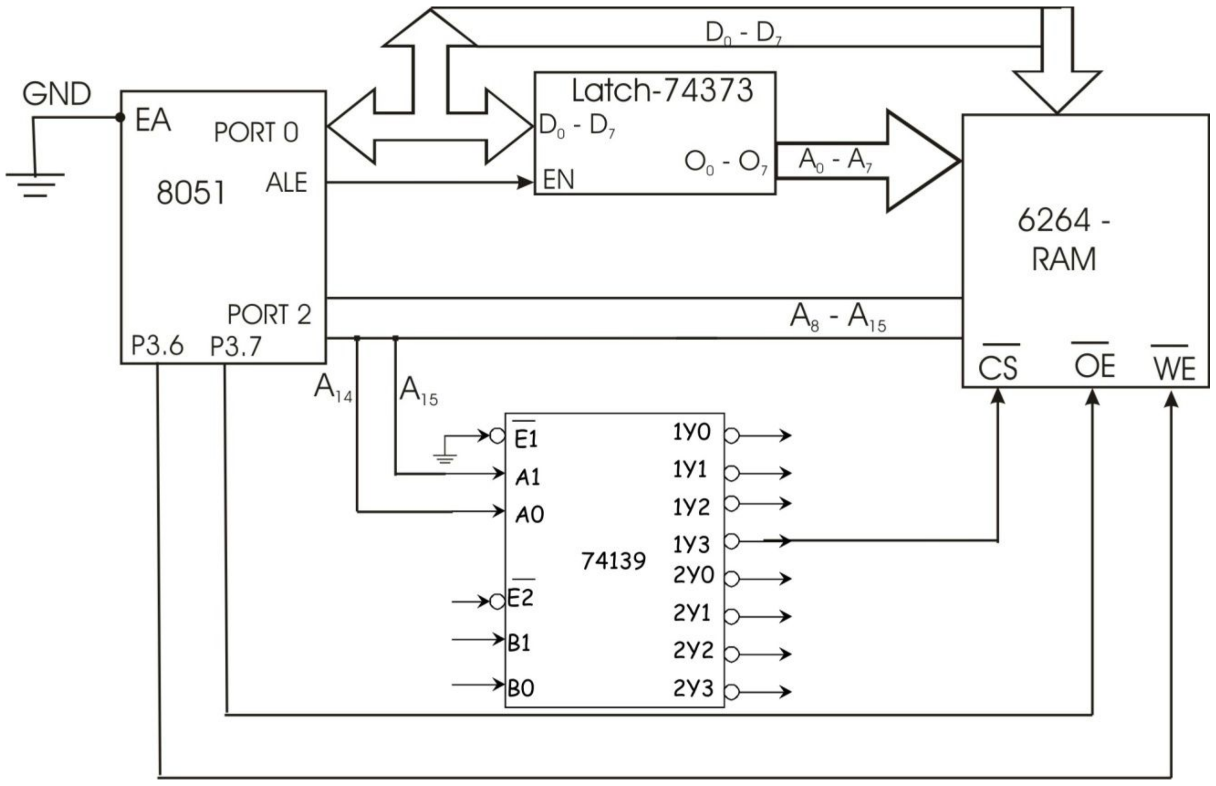


Fig. 18 Connecting RAM Chip 6264 to 8051