ARITHMETIC OPERATIONS

The 8051 supports number of arithmetic and logical operations. ADDITION, SUBTRACTION, MULTIPLICATION, DIVISION, INCREMENTING AND DECREMENTING and DECIMAL ADJUST can be used to perform arithmetic operations. The arithmetic opcodes are -

MNEMONIC	OPERATION
INC destination	Increment destination by 1
DEC destination	Decrement destination by 1
ADD destination, source	Add source to destination
ADDC destination , source	Add source to destination with carry.
SUBB destination, source	Subtract with barrow, source from destination
MUL AB	Multiply the contents of register A and B
DIV AB	Divide the contents of register A by the contents of register B
DA A	Decimal Adjust the A register

Incrementing and Decrementing:

The instructions Incrementing and Decrementing, increments/ decrements the contents of a register or a memory location by 1.The list of increment and decrement operations is given below.

MNEMONIC	OPERATION
INC A	Add 1 to accumulator
INC Rp	Add 1 to register Rr
INC @Rp	Add 1 to contents of the direct memory address
INC add	Add 1 to contents of memory address in Rp
INC DPTR	Add 1 to 16 bit contents of DPTR
DEC A	Subtract 1 from accumulator
DEC Rr	Subtract 1 from register Rr
DEC add	Subtract 1from contents of the direct memory address
DEC @Rp	Subtract 1 from contents of memory address in Rp

Note:

- There is no DEC DPTR instruction.

Addition (Without Carry):

All addition is done with the A register as the destination of the result. All addressing modes may be used for the source. The following table lists the addition mnemonics.

MNEMONIC	OPERATION
ADD A, #n Eg: ADD A, #3C	Add A and the immediate number n ; put the sum in A
ADD A,#Rr Eg: ADD A, R4	Add A and register Rr; put the sum in A
ADD A,add Eg: ADD A, 3C	Add A and address contents; put the sum in A
ADD A,@Rp Eg: ADD A,@R1	Add A and the contents of the address in Rp; put the sum in A

Addition (With Carry):

If carry is to be added to the higher order bytes of the addition operation, ADDC instruction is used. The following table lists the ADD instruction with carry mnemonics.

MNEMONIC	OPERATION
ADDC A, #n Eg: ADDC A, #45	Add A and the immediate number n and the C flag ; put the sum in A
ADD C A, Rr Eg: ADDC A, R3	Add A and register Rr and C flag; put the sum in A
ADDC A,add Eg: ADDC A, 45	Add A and address contents and C flag ; put the sum in A
ADDC A,@Rp Eg: ADDC A,@R0	Add A and the contents of the address in Rp and C flag ; put the sum in A

<u>In all the above instructions the result is stored in A</u>

<u>Subtraction:</u>

Subtraction can be done by taking the two`s complement of the number to be subtracted (subtrahend), and adding it to another number (minuend). **<u>Register A is the destination address for the subtraction</u>**. All addressing modes may be used for source address. The commands treat the Carry flag as a borrow flag and always subtract the carry flag as part of the operation.

The following table lists the subtraction mnemonics.

MNEMONIC	OPERATION
SUBB A, #n Eg: SUBB A, #55	Subtract the immediate number n and the C flag from A ; put the result in A
SUBB A, Rr Eg: SUBB A, R6	Subtract register Rr and C flag from A ; put the result in A
SUBB A,add Eg: SUBB A, 55	Subtract the address contents and C flag from A ; put the sum in A
SUBB A,@Rp Eg: SUBB A,@1	Subtract the contents of the address in Rp and C flag from A ; put the sum in A

<u>In all the above instructions the result is stored in A</u>

<u>Multiplication:</u>

Multiplication operation uses registers A and B as both source and destination addresses for the operation. The number in register A is multiplied by the number in register B as indicated in the following instruction.

MNEMONIC	OPERATION
MUL AB	Multiply A by B; put the lower order byte of the product in A, and higher order byte of the product in B

<u>Division:</u>

Division operation uses registers A and B as both source and destination addresses for the operation. The number in register A is divided by the number in register B as indicated in the following instruction.

MNEMONIC	OPERATION
DIV AB	Divide A by B; put the integer part of the quotient in register A and integer part of the reminder in B

Decimal Arithmetic:

Sometimes it becomes necessary to used decimal numbers 0-9 than hex or binary numbers. These are represented in BCD numbers.

Unpacked BCD:

Unpacked BCD is the term used to represent decimal numbers with eight bits. The upper nibble is zero and the lower nibble represents the BCD number.

Eg; 0000 0001 represents unpacked BCD 1 and 0000 1001 represents the unpacked BCD 9. This is used to represent the numbers from 0-9.

Packed BCD:

In packed BCD, a single byte has a two BCD numbers in it. One is the lower 4-bits and another is the upper 4-bits.

Eg: 0100 1001 represents packed BCD of 49 and 1000 0001 represents the packed BCD of 81.

The 8051 supports an instruction "DA" (Decimal Adjust for addition) which is used to add two packed BCD numbers. The instruction is as follows

MNEMONIC	OPERATION
DA A	Adjust the sum of two packed numbers for decimal addition. The adjusted number is in A

LOGICAL OPERATIONS

The Boolean operators are – AND, OR, NOT and XOR. The two data levels –byte or bit at which the Boolean instructions operate are as shown in the following table

Boolean operator	8051 mnemonic
AND	ANL (logical AND)
OR	ORL (logical OR)
XOR	XRL (logical Exclusive OR)
NOT	CPL (complement)

Byte level logical operations:

Byte level logical operations involve each individual bit of a source byte operating on the same bit positions in the destination byte. The result is put in the destination byte. The source byte is not changed.

The byte level logical operations use all four addressing modes for the source of data byte. **A register or a direct address in internal RAM is the destination of the logical operation result**. These operations are called byte level Boolean operation because the entire byte is affected.

The following are the byte level Boolean Operations.

MNEMONIC	OPERATION
ANL A,#n Eg: ANL A,#8C	AND each bit of A with the same bit of immediate number n; put the results in A
ANL A ,add Eg: ANL A, 7C	AND each bit of A with the same bit of direct RAM address; put the results in A
ANL A,Rr Eg: ANL A, R3	AND each bit of A with the same bit of register Rr; put the results in A
ANL A, @Rp Eg: ANL A, @R1	AND each bit of A with the same bit of the contents of the RAM address contained in Rp; put the results in A
ANL add , A Eg:ANL 27, A	AND each bit of A with the direct RAM address; put the results in direct RAM address.
ANL add,#n Eg: ANL 4E,#5	AND each bit of RAM address with the same bit in the number n ; put the results in the direct RAM address.
ORL A,#n Eg: ORL A,#8C	OR each bit of A with the same bit of immediate number n; put the results in A
ORL A ,add Eg: ORL A, 7C	OR each bit of A with the same bit of direct RAM address; put the results in A
ORL A,Rr Eg: ORL A, R3	OR each bit of A with the same bit of register Rr; put the results in A
ORL A, @Rp Eg: ORL A, @R1	OR each bit of A with the same bit of the contents of the RAM address contained in Rp; put the results in A
ORL add , A Eg: ORL 27, A	OR each bit of A with the direct RAM address; put the results in direct RAM address.
ORL add,#n Eg: ORL 6F,#n	OR each bit of RAM address with the same bit in the number n ; put the results in the direct RAM address.
XRL A,#n Eg: XRL A,#58	XOR each bit of A with the same bit of immediate number n; put the results in A
XRL A ,add Eg: XRL A ,68	XOR each bit of A with the same bit of direct RAM address; put the results in A
XRL A,Rr Eg: XRL A,R2	XOR each bit of A with the same bit of register Rr; put the results in A
XRL A, @Rp Eg: XRL A, @R1	XOR each bit of A with the same bit of the contents of the RAM address contained in Rp; put the results in A
XRL add , A Eg: XRL 80, A	XOR each bit of A with the direct RAM address; put the results in direct RAM address.
XRL add,#n Eg: XRL 8C,#55	XOR each bit of RAM address with the same bit in the number n; put the results in the direct RAM address.
CLR A	Clear each bit of register A to 0
CPL A	Complement each bit of A; every 1 becomes 0, and each 0 becomes 1.

Note:

- The destination of the result of the logical operation is either the register A or the direct address in the internal RAM.

Bit level logical operations:

Bit level operations involve individual bits found in the internal RAM and certain SFRs that may be addressed either by direct address or by individual bit addresses. The following are the bit address.

Byte addresses	Bit addresses	Byte addresses	Bit addresses
Internal RAM 20	00 – 07	2C	60 – 67
21	08 – 0F	2D	68 – 6F
22	10 – 17	2E	70 – 77
23	18 – 1F	2F	78 – 7F
24	20 – 27	SFRs A - 0E0	0E0 – 0E7
25	28 – 2F	B - 0F0	0F0 – 0F7
26	30 – 37	IE - 0A8	0A8 – 0AF
27	38 – 3F	IP - 0B8	0B8 – 0BF
28	40 – 47	P0 - 80	80 – 87
29	48 – 4F	P1 - 90	90 – 97
2A	50 – 57	P2 - 0A0	0A0 – 0A7
2B	58 – 5F	P3 - 0B0	0B0 – 0B7

The following table lists the bit level Boolean Operations;

MNEMONIC	OPERATION
ANL C,b	AND C and the addressed bit b; put the result in C
ANL C, /b	AND C and complement of the addressed bit b; put the result in C
ORL C, b	OR C and the addressed bit b; put the result in C
ORL C, /b	OR C and complement of the addressed bit b; put the result in C
CPL C	Complement the C flag
CPL b	Complement the addressed bit
CLR C	Clear the C flag (C = 0)
CLR b	Clear the addressed bit
MOV C, b	Copy the addressed bit to C flag
MOV b, C	Copy the C flag to the addressed bit
SETB C	Set the carry flag (C = 1)
SETB b	Set the addressed bit to 1

Note:

- In the entire above, no flag except carry flag is affected.
- If the destination bit is a port bit, the SFR latch bit is affected, not the pin.
- Only SFRs which are bit addressable can be used in bit level operations.
- ANL C, /b and ORL C, /b do not alter the addressed bit b.

Rotate and Swap operations:

The ability to rotate data is useful for inspecting bits of a byte. The A register can be rotated one bit position to the left or right with or without carry flag. If C flag is not used then the rotate operation involves 8 bits. If the carry flag is used, the rotate operation uses 9 bits.

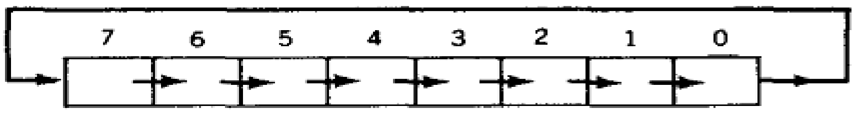
The SWAP operation interchanges the upper and lower nibbles of the Accumulator.

In ROTATE and SWAP operations, the A register is used.

MNEMONIC	OPERATION
RL A	Rotate the A register one bit position to the left.
RLC A	Rotate the A register and the carry flag as a ninth bit, one bit position to the left.
RR A	Rotate the A register one bit position to the right
RRC A	Rotate the A register and the carry flag as the ninth bit, one bit position to the right.
SWAP A	Interchange the nibbles of A register.

RR A: Rotate Accumulator right.

The 8-bits of the accumulator are rotated right one bit. A0 is placed to A7 and every other bit moved right by 1 bit.

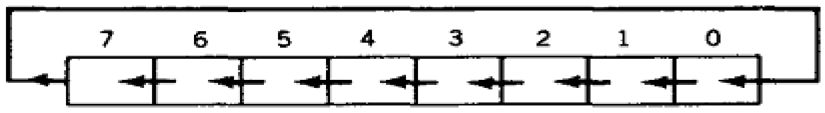


RR A

Eg: If A = 0011 0110 then RR A = 0001 1001

RL A: rotate Accumulator left:.

The 8-bits of the accumulator are rotated right one bit. A7 is placed to A0 and every other bit moved left by 1 bit.

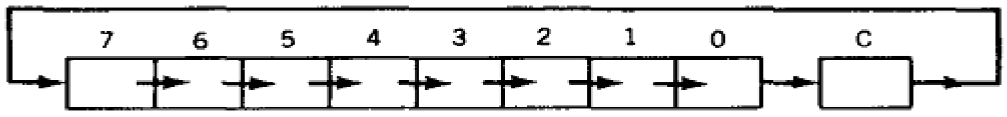


RL A

Eg: If A = 0011 0110 then RLA = 0110 1100

<u>RRC A: rotate Accumulator right through carry :</u>

In RRC A, accumulator bits are rotated right, the LSB A0 moves into carry and carry moves to A7. It is considered as 9 bit rotation including carry.

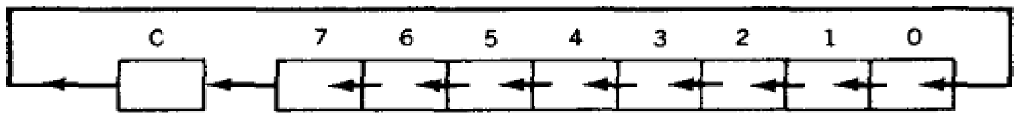


RRC A Carry Flag

Eg: If A = 0011 0110 and CY = 1, then RRC A = 1001 1011 CY = 0

<u>RLC A: rotate Accumulator left through carry :</u>

In RLC A, accumulator bits are rotated left, the MSB A7 moves into carry and carry moves to A0. It is considered as 9 bit rotation including carry.

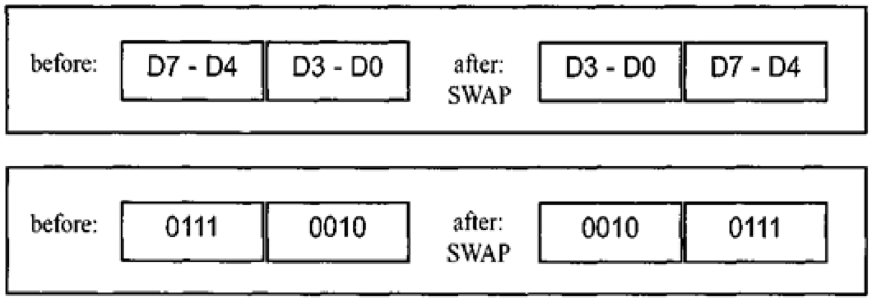


Carry Flag RLC A

Eg: If A = 0011 0110 and CY = 1, then RLC A = 0110 1101 and CY = 0

<u>SWAP operation:</u>

The SWAP operation interchanges the upper and lower nibbles of the Accumulator.



8051 PROGRAMMING

ADDRESSING MODES

The CPU can access data in various ways. The data could be in a register, or in memory, or be provided as an immediate value. **These various ways of accessing data are called *addressing modes*.** . In 8051 there are five distinct addressing modes. They are -

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

IMMEDIATE ADDRESSING MODE

In this mode, the data is made part of the opcode. The source operand is a constant and appears immediately after the opcode. The data is prefixed with **"#"** sign, to indicate it is the operand itself.

This addressing mode can be used to load information into any of the registers, including the DPTR register and Ports. Although the DPTR register is 16-bit, it can also be accessed as two 8-bit registers, DPH and DPL, where DPH is the high byte and DPL is the low byte.

Mnemonic	Operation
MOV Rr.#n Eg: MOV R3,#2A MOV B, #22	Copy the 8 bit number n into register Rr
MOV A.#n Eg: MOV A, #20H	Copy the 8 bit number n into Accumulator register
MOV DPTR,#nn Eg: MOV DPTR, #32FAH MOV DPL, #FF	Copy the 16 bit number nn into DPTR register

R_r can be any of the registers R0 to R7 in the selected bank

REGISTER ADDRESSING MODE

This involves the use of registers to hold the data to be manipulated. Both the source and destination are registers and their sizes should match.

Mnemonic	Operation
MOV A, Rr Eg: MOV A, R0 ADD A, R7	Copy data from register Rr into A
MOV Rr, A Eg: MOV R5, A MOVR3, A	Copy data from register A to register Rr

NOTE:

- ALL NUMBERS MUST START WITH A DECIMAL NUMBER 0-9
- REGISTER TO REGISTER MOVES OCCUR BETWEEN REGISTERS A AND R0-R7

- MOV R1,R2 IS INVALID INSTRUCTION

DIRECT ADDRESSING MODE:

In this mode the direct address of memory location is provided in instruction to fetch the operand. Only internal RAM and SFR's address can be used. All 128 bytes of internal RAM and the SFR may be addressed directly using single byte address assigned to each RAM location and each special function register. We know that

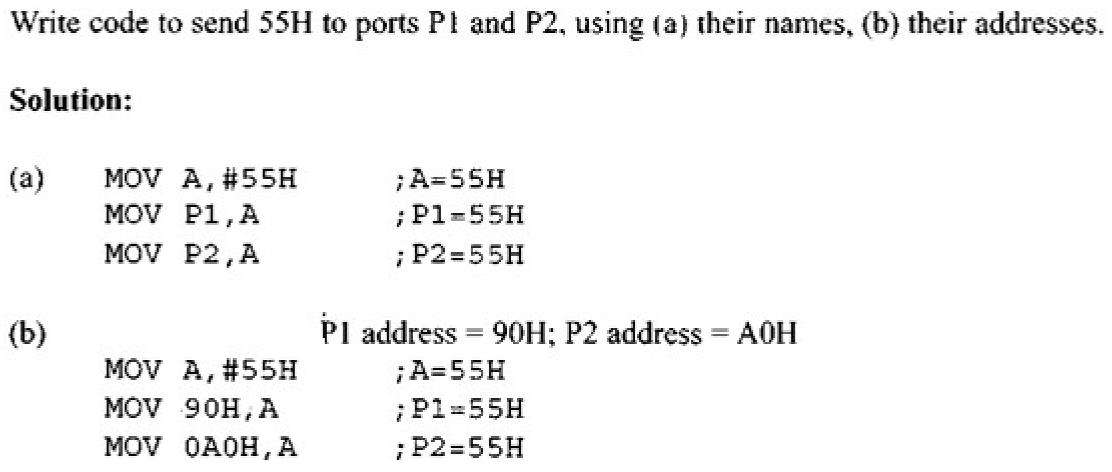
- RAM locations 00 – 1FH are assigned to the register banks and stack.
- RAM locations 20 – 2FH are bit addressable
- RAM locations 30 – 7FH are available for general purpose RAM.
- Some addresses between 80 – FFH are uniquely assigned to SFR.

The SFRs addresses are;

SFR	ADDRESS	SFR	ADDRESS
A	0E0	PSW	0D0
B	0F0	SBUF	99
DPL	82	SCON	98
DPH	83	SP	81
IE	0A8	TCON	88
IP	0B8	TMOD	89
P0	80	TH0	8C
P1	90	TL0	8A
P2	0A0	TH1	8D
P3	0B0	TL1	8B
PCON	87		

The structure of direct addressing modes is as shown below.

MNEMONIC	OPERATION
MOV A, add Eg: MOV A, 30H;	Copy data from direct add to register A Copy data from direct address 30 to register A;
MOV add, A Eg: MOV 23H, A	Copy data from A to direct address add Copy data from A to direct address address 23;
MOV Rr, add Eg: MOV R1, 75H	Copy data from direct add to register Rr Copy data from direct address 75 to register R1;
MOV add, Rr Eg: MOV 75H , R1	Copy data from register Rr direct address add Copy data from R1 to direct address 75 ;
MOV add, #n Eg: MOV 52H ,# 0FH	Copy immediate data byte n to direct address add Copy immediate data byte 0F to direct address 52;
MOV add1, add2 Eg: MOV 45H, 36H	Copy data from direct address add2 to direct address add1 Copy data from direct address 36 to direct address 45;


Write code to send 55H to ports P1 and P2, using (a) their names, (b) their addresses.

Solution:

```
(a)     MOV A,#55H      ;A=55H
        MOV P1,A        ;P1=55H
        MOV P2,A        ;P2=55H

(b)                     P1 address = 90H; P2 address = A0H
        MOV A,#55H      ;A=55H
        MOV 90H,A       ;P1=55H
        MOV 0A0H,A      ;P2=55H
```

DATA TRANSFER WITH STACK:

The stack is a section of RAM used by CPU for storage of information temporarily. This is needed since we have limited number of registers. The register used to access the stack is the stack pointer SP. By default, when power is switched on, the SP is initialized at 07H, which is the address of R7 of Bank 0. There are two data functions *PUSH & POP* associated with Stack.

PUSH OPCODE:

A PUSH opcode copies data from the source address to the stack. SP is incremented by 1 before the data is copied to the internal RAM location contained in SP. The data is stored from low address to high address in the internal RAM. The stack grows up in memory as it is PUSHed. When the SP reaches FFH it “rolls over” to 00H and the data is lost.

The Mnemonic is

```
PUSH  add  ; Increment  SP. Copy the contents of add to the internal RAM address in SP
```

For eg:

MOV R3, # 33H

MOV R2, #22H
PUSH 03H
PUSH 02H

In this example the SP is initialized at 07H. The first two instructions load the data 33 and 22 into registers R3 and R2. . Stack is changed as shown below.

0B	
0A	
09	
08	

SP = (

0B	
0A	
09	
08	33

0B	
0A	
09	22
08	33

P = 09

POP OPCODE:

A POP opcode copies from the stack to the destination address. SP is decremented by 1 after data is copied from stack RAM address to direct destination. The data is retrieved in the same manner as it was stored.

The Mnemonic is

POP add; Copy the contents of the internal RAM address in SP to add. ; Decrement SP

E.g.: POP 02
POP 01
POP 00

0B	5B
0A	5A
09	59
08	58

Original
SP = 0B

0B	
0A	5A
09	59
08	58

After POP 02
R2 = 5B
SP = 0A

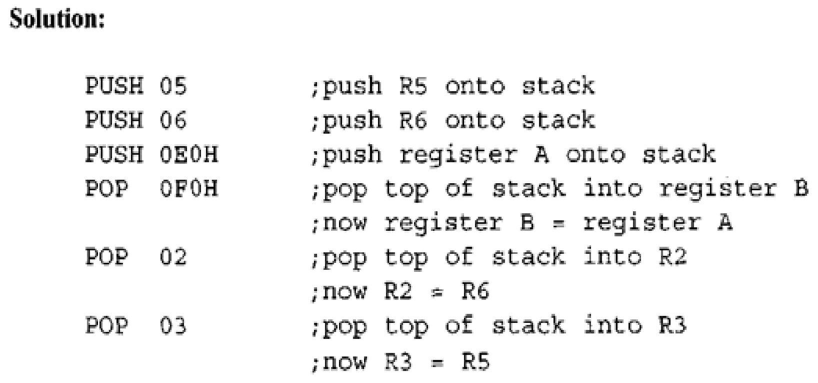
0B	5B
0A	5A
09	59
08	58

After POP 01
R1 = 5A
SP = 09

0B	5B
0A	5A
09	59
08	58

After POP 00
R0 = 59
SP = 08

Program
Show the code to push R5, R6, and A onto the stack and then pop them back them into R2, R3, and B, where register B = register A, R2 = R6, and R3 = R5.


Solution:

```
PUSH 05        ;push R5 onto stack
PUSH 06        ;push R6 onto stack
PUSH 0E0H      ;push register A onto stack
POP  0F0H      ;pop top of stack into register B
               ;now register B = register A
POP  02        ;pop top of stack into R2
               ;now R2 = R6
POP  03        ;pop top of stack into R3
               ;now R3 = R5
```

REGISTER INDIRECT ADDRESSING MODE:

In the register indirect addressing mode, a register is used as a pointer to the data. If the data is inside the CPU, only registers R0 and Rl are used for this purpose. In other words, R2 – R7 cannot be used to hold the address of an operand located in RAM when using this addressing mode.

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

Notice that R0 (as well as Rl) is preceded by the "@" sign. In the absence of the "@" sign, MOV will be interpreted as an instruction moving the contents of register R0 to A, instead of the contents of the memory location pointed to by R0.

MNEMONIC	OPERATION
MOV @Rp , #n Eg: MOV @R1, #35H	Copy the immediate byte 'n' to the address in Rp. Copy the number 35H to the address in R1;
MOV @Rp, add Eg: MOV @R1 , #35H;	Copy the contents of add to the address in Rp Copy the number 35 to the address in R1;
MOV @Rp, A Eg: MOV @R1,A ;	Copy the data in A to the address in Rp Copy the data in A to the address in R1;
MOV add @Rp Eg: MOV 40H , @R0 ;	Copy the contents of address in Rp to add Copy the contents of address in R0 to address 40;
MOV A, @Rp Eg: MOV A,@R0 ;	Copy the contents of the address in Rp to A Copy the contents of the address in R0 to A;

Sometimes it becomes necessary to access the pre-programmed data. This data is permanent and is stored in ROM locations. This data can be accessed using indirect addressing mode between the register A and either with DPTR or PC. The contents of A are added to the pointer to get the address of the data. The data is placed in A. hence the old data in A is lost. The syntax is as follows.

The syntax is as follows.

MNEMONIC	OPERATION
MOVC A,@A +DPTR	Copy the code byte , found in ROM address formed by adding A and DPTR to A
MOVC A,@A +PC	Copy the code byte , found in ROM address formed by adding A and PC to A

Indexed addressing mode and MOVX instruction

The 8051 has 64K bytes of memory space for data storage. This data memory space is referred to as ***external memory***. It is accessed by the MOVX instruction. . An 'X' is added to the MOV mnemonics to indicate that the data move is external to 8051

MNEMONIC	OPERATION
MOVX A,@Rp Eg: MOVX A,@R0	Copy the contents of the external address in Rp to A
MOVX A,@DPTR	Copy the contents of the external address in DPTR to A
MOVX @Rp, A	Copy data from A to the external address in Rp

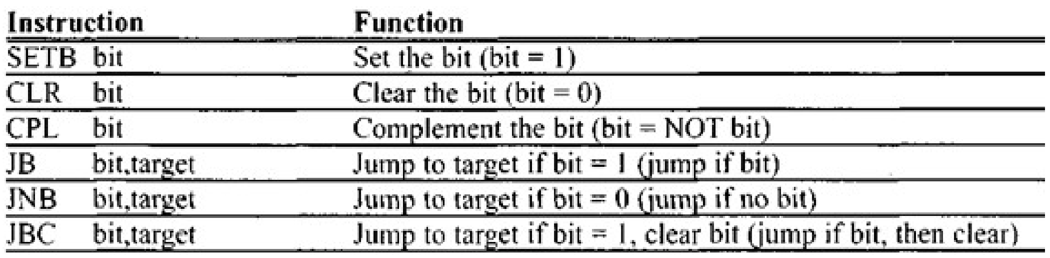
Eg: MOVX @R1,A	
MOVX @DPTR , A	Copy data from A to the external address in DPTR

One of the important features of the 8051 is the ability to access the registers, RAM, and I/O ports in bits instead of bytes.

Bit-addressable RAM

The bit-addressable RAM locations are 20H to 2FH. These 16 bytes provide 128 bits of RAM bit-addressability. They are addressed as 0 to 127 (in decimal) or 00 to 7FH. In orderto access these 128 bitsof RAM locations andother bit-addressablespace of 8051 individually, single-bit instructions such as SETB is used

Single-Bit Instructions



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

DATA EXCHANGE:

MOV, PUSH and POP instructions involve copying the data found in source address to the destination address; the original data in the source is not changed. Exchange instructions move data from source to destination and from destination to source. In effect the data is swapped between two locations. All addressing modes except immediate addressing mode may be used. The mnemonic is **XCH**

e.g.;

MNEMONIC	OPERATION
XCH A, Rr Eg: XCH A, R2	Exchange data bytes between Rr and A
XCH A, add Eg: XCH A, 4A	Exchange data bytes between add and A
XCH A, @Rp Eg: XCH A, @R1	Exchange data bytes between A and address in Rp
XCHD A, @Rp Eg: XCHD A, @R0	Exchange lower nibble between A and address in Rp

NOTE:

- All exchanges are internal to 8051
- All exchanges use register A
- When using XCHD, the upper nibble of A and upper nibble of address location in Rp do not change.

MICRONTROLLERS

A Microcontroller is a computer present in a single integrated circuit (IC) which is dedicated to perform a particular task and execute. It contains memory, programmable input/output peripherals as well a processor.

Difference between Microprocessor and Microcontroller

Sl. No	**Microprocessor**	**Microcontroller**
1	CPU is stand alone, RAM,ROM ,I/O devices and timers are separate and interfaced with CPU	CPU, RAM, ROM,I/O devices and TIMERS are all located on a single chip.
2	Designer can decide on the amount of RAM, ROM and I/O ports.	Fixed amount of RAM,ROM and I/O ports
3	Expensive applications	Applications in which cost, space and power are critical
4	Versatile and general purpose	Not very versatile
5	Program memory and data memory are same	Uses different program memory and data memory.
6	Large number of instructions with flexible addressing modes	Limited number of instructions with few addressing modes
7	Very few instructions which have bit handling capability	Many instructions with bit handling capability
8	System cost is more	System cost is less
9	Clock frequency > 1 G Hz	Clock frequency 10-20 M Hz

Important features of 8051:

- 4Kb ROM
- 128 bytes RAM
- Four 8 bits I/O ports
- Two 16 bit timers
- Serial interface
- 64K external code memory space
- 64 K data memory space.

ARCHITECTURE OF 8051 MICROCONTROLLER

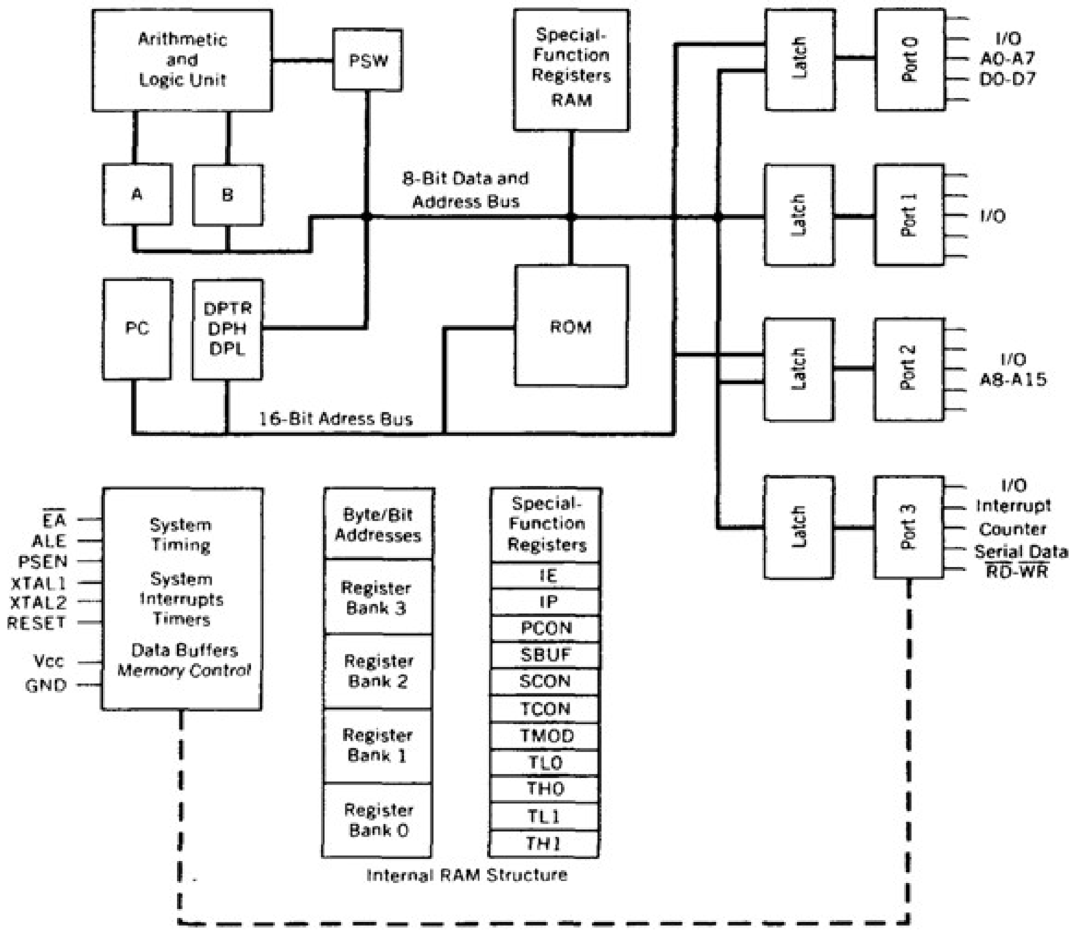
The 8051 architecture consists of these specific features:

- Eight bit CPU with register A (Accumulator) and B.
- 16-bit program counter (PC) and Data pointer (DPTR).
- 8-bit Program Status Word (PSW).
- 8-bit Stack pointer (SP).
- Internal ROM 0 to 4K.
- Internal RAM of 128 bytes:
    - 4-Register Banks, each containing 8-registers
    - 16 bytes, which may be addressed at the bit level.
    - 8 bytes of general-purpose data memory.
- 32 Input/output pins arranged as four 8-bit ports: P0-P3.
- Two 16-bit Timer/Counters: T0 and T1
- Full duplex serial data receiver/transmitter: SBUF.

- Control Registers: TCON, TMOD, SCON, PCON, IP and IE.
- 2 external and 3 internal interrupt sources.
- Oscillator and Clock circuits.

The architectural block diagram of the microcontroller is as shown below. The architecture can be explained under the following headings.

1. Registers of microcontroller
2. Memory
3. Timers/counters
4. Interrupts.



__REGISTERS IN 8051:__

The 8051 has general purpose or working registers, Stack Pointer, Program Counter in addition to the CPU registers. There are special function registers also.

__Program Counter (PC)__

The program counter is a 16 bit register. It points to the address of the next instruction to be executed. Instruction opcodes are fetched from the program memory locations (ROM) addressed by the program counter. After the CPU fetches the op-code, the PC is automatically incremented to point to the next instruction. The PC is the only register which does not have an internal address. The PC is 16 bits wide i.e. it can access program address from 0000 to FFFF.

Data pointer (DPTR):

The DPTR is a 16 bit register, made up of two 8 bit registers- DPH and DPL. The higher byte is referred to as DPH and the lower byte is referred to as DPL. The data pointer is used for addressing external memory. With 16 bit DPTR, a maximum of 64 k off-chip data memory and program memory can be accessed.

General purpose Registers (A and B CPU registers):

The 8051 contains 34 general purpose or working registers. Two of these registers A and B hold results of many instructions, particularly math and logical operations. The other 32 registers are arranged as part of internal RAM in four banks B0-B3, of eight registers.

Accumulator (A):

This is one of the CPU register. It is 8 bit register used by all arithmetic and logical instructions. This stores one of the operands of the instruction before execution. It also stores the result after the execution of an instruction. Access to accumulator is faster than the main memory.

B-register:

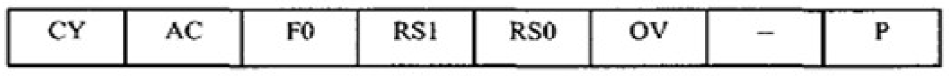
This is another register of CPU. It is an 8 bit register. It is being used for arithmetic operations of multiplication and division. It is available as general purpose register when it is not being used by multiplication and division operations.

Registers R0 through R7:

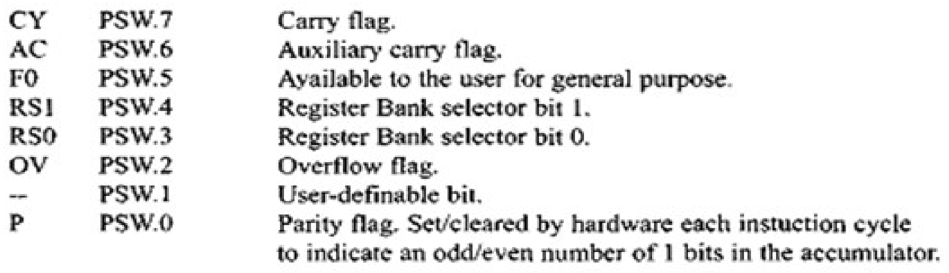
These 8 registers are used as scratch pad registers. There are 4 register banks each containing R0 through R7 registers. Each of these registers is 8 bit wide. Any register bank can be selected by appropriate setting of bits in PSW. These register banks are located in the on-chip RAM.

Flags and Program Status Word (PSW):

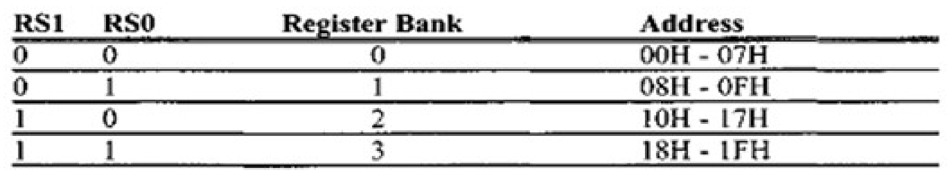
Flags are 1-bit registers provided to indicate the results of certain conditions like carry, parity, sign etc., of a program. Other instructions can test the conditions of the flags and make decisions based on flags status. To address the flags conveniently they are grouped inside the Program Status Word (PSW) The bit format of PSW is as shown below



CY	AC	F0	RS1	RS0	OV	–	P



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



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

8051 has 4 math flags that respond to the output of math operations and 3 general purpose flags. The math flags are – Carry (C), Auxiliary Carry (AC), Overflow (OV) and Parity (P).

The general purpose flags are – F0, GF0 and GF1 that may be used by the programmer to record some event in the program.

Carry flag:

Carry flag is set when there is a carry out of 7th bit of the result due to certain arithmetic or logical operations. E.g.:11000010 +10101010 = 1] 0110110. This will set the carry flag.

Auxiliary carry flag:

Auxiliary carry flag is set when there is a carry out of 3rd bit during addition or subtraction operation and otherwise cleared. This is used in BCD arithmetic.

Overflow flag:

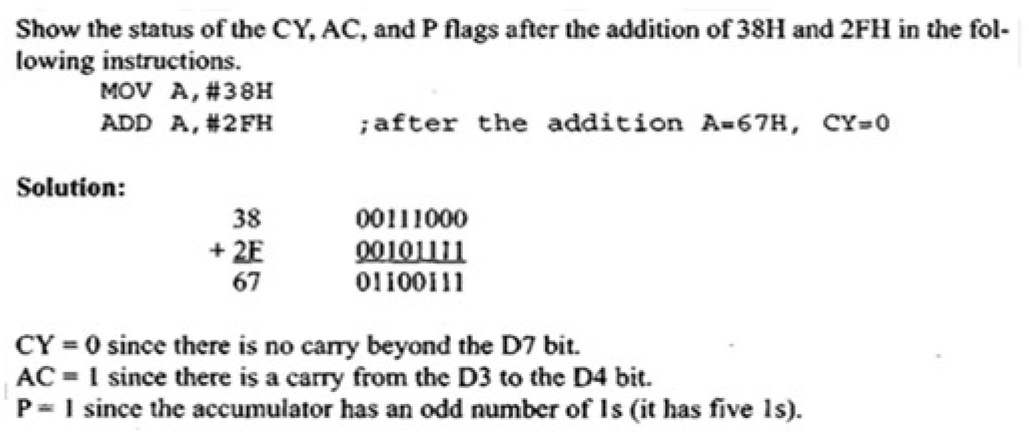
Overflow is set as a result of an arithmetic operation provided there is a carry out of bit 6, but not out of bit 7 OR a carry out of bit 7 but not out of bit 6.

E.g.: 11000010 + 10101010 = 01101100. Here carry is generated out of bit 7 but not out of bit 6. HenceOV flag is set.

Parity flag:

Parity flag indicates the number of 1`s in accumulator. If there are odd number of 1`s in accumulator, then the odd parity sets the parity flag to 1. The parity flag is 0, for even parity

Problems:



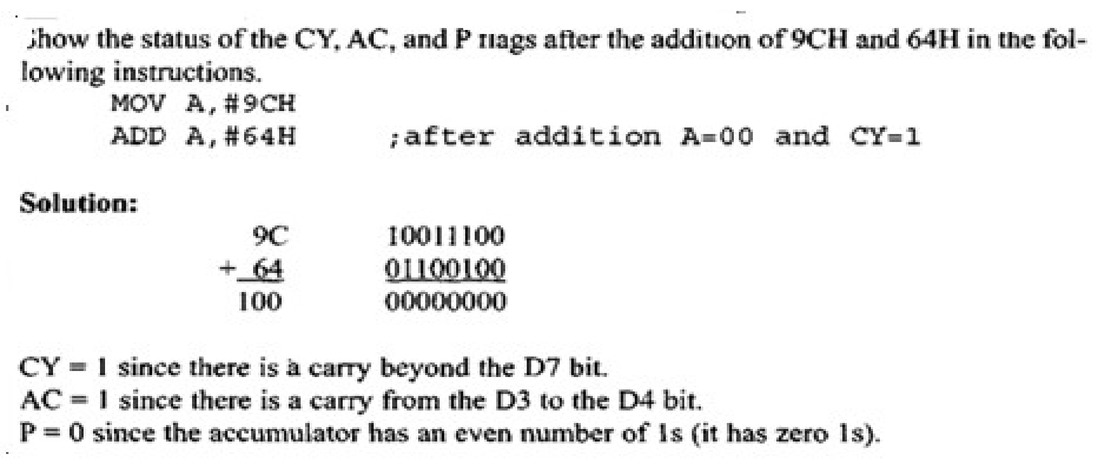
Show the status of the CY, AC, and P flags after the addition of 38H and 2FH in the following instructions.

```
      MOV  A,#38H
      ADD  A,#2FH      ;after the addition A=67H, CY=0
```

Solution:

```
            38        00111000
          + 2F        00101111
            67        01100111
```

CY = 0 since there is no carry beyond the D7 bit.
AC = 1 since there is a carry from the D3 to the D4 bit.
P = 1 since the accumulator has an odd number of 1s (it has five 1s).



Show the status of the CY, AC, and P flags after the addition of 9CH and 64H in the following instructions.

```
      MOV  A,#9CH
      ADD  A,#64H      ;after addition A=00 and CY=1
```

Solution:

```
            9C        10011100
          + 64        01100100
           100        00000000
```

CY = 1 since there is a carry beyond the D7 bit.
AC = 1 since there is a carry from the D3 to the D4 bit.
P = 0 since the accumulator has an even number of 1s (it has zero 1s).

Show the status of the CY, AC, and P flags after the addition of 88H and 93H in the following instructions.

```
          MOV A,#88H
          ADD A,#93H      ;after the addition A=1BH,CY=1
```

Solution:

```
          88        10001000
        + 93        10010011
         11B        00011011
```

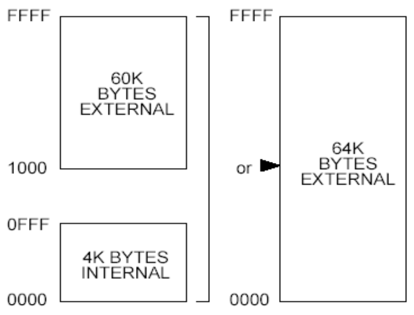
CY = 1 since there is a carry beyond the D7 bit.
AC = 0 since there is no carry from the D3 to the D4 bit.
P = 0 since the accumulator has an even number of 1s (it has four 1s).

MEMORY ORGANIZATION

Internal ROM

The 8051 has 64 K external data memory, 256 bytes of internal data memory and 64 K program memory. The program memory map of 8051 is as shown below.

The 80C51 Program Memory.



The 8051 is organized so that the data memory and program code memory can be in two entirely different physical memory entities. The 64 K program memory space is divided into internal and external memory. Internal program code contained in an internal ROM has the address range from 0000h to 0FFFh.

Program addresses higher than 0FFFh which exceed the internal ROM capacity, will cause the 8051 to automatically fetch code bytes from the external memory portion i.e., 1000H through 0FFFFH

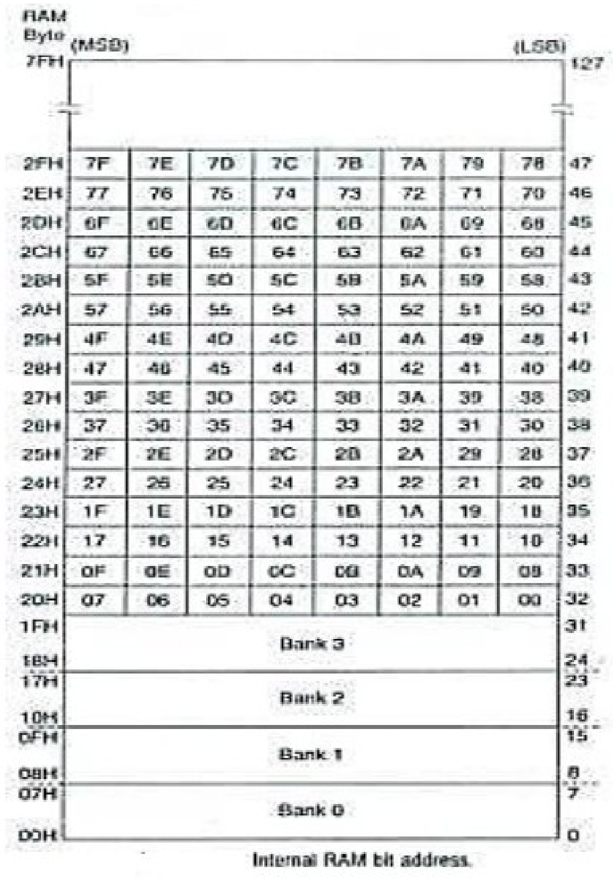
Internal RAM

The internal data memory of 8051 is 256 bytes, which is divided into two parts. The lower 128 bytes (00H through 7FH) called as the internal data RAM and the upper 128 bytes (80H through FFH)are called SFRs.

The 128-byte internal RAM is organized into three distinct areas.

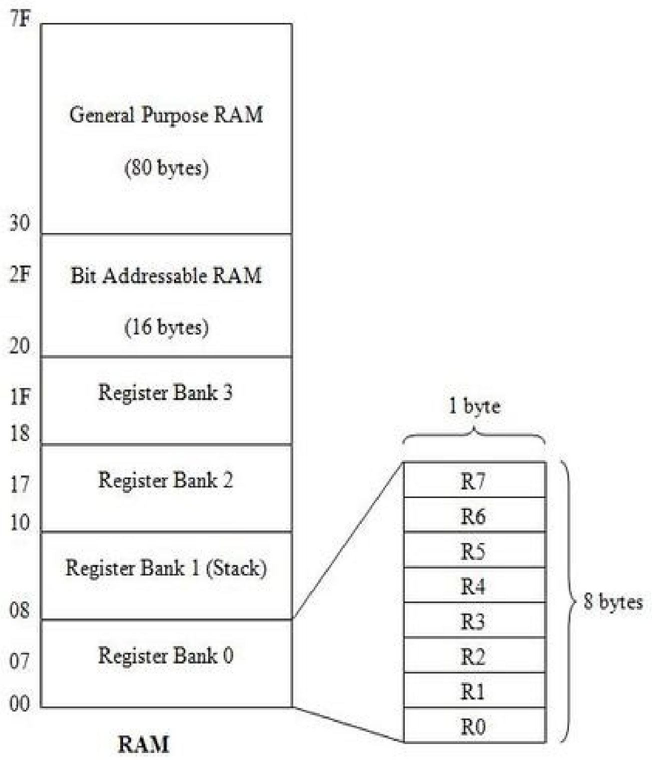
- Register banks
- Bit addressable RAM
- General purpose RAM

32 bytes from address 00h to 1Fh are made up of 32 working registers. These registers are organized as four banks of 8 registers each. The four register banks are numbered 0 to 3 and registers named R0 to R7. Each register can be addressed by name (when bank is selected) or by its RAM address. Bits RS0 and RS1 in the PSW determine which bank of registers is currently in use. Bank 0 is selected on reset.



RAM Byte	(MSB)							(LSB)	
7FH									127
2FH	7F	7E	7D	7C	7B	7A	79	78	47
2EH	77	76	75	74	73	72	71	70	46
2DH	6F	6E	6D	6C	6B	6A	69	68	45
2CH	67	66	65	64	63	62	61	60	44
2BH	5F	5E	5D	5C	5B	5A	59	58	43
2AH	57	56	55	54	53	52	51	50	42
29H	4F	4E	4D	4C	4B	4A	49	48	41
28H	47	46	45	44	43	42	41	40	40
27H	3F	3E	3D	3C	3B	3A	39	38	39
26H	37	36	35	34	33	32	31	30	38
25H	2F	2E	2D	2C	2B	2A	29	28	37
24H	27	26	25	24	23	22	21	20	36
23H	1F	1E	1D	1C	1B	1A	19	18	35
22H	17	16	15	14	13	12	11	10	34
21H	0F	0E	0D	0C	0B	0A	09	08	33
20H	07	06	05	04	03	02	01	00	32
1FH–18H	Bank 3								31–24
17H–10H	Bank 2								23–16
0FH–08H	Bank 1								15–8
07H–00H	Bank 0								7–0

Internal RAM bit address.



- There are 4 register banks from 00 to 1F each containing R0 through R7 registers. Each of these registers is 8 bit wide. At any time only one register bank can be selected by appropriate setting of bits in PSW.
- A bit addressable area of 16 bytes occupies RAM byte addresses 20h to 2Fh, forming a total of 128 addressable bits. An addressable bit may be specified by the bit address of 00h to 7Fh, or 8 bits from any byte address from 20h to 2Fh.

- A general-purpose RAM area above the bit area from 30h to 7Fh addressable as byte.

Special Function Registers (SFR):

The 8051 operations that do not use the internal 128-byte RAM addresses from 00h to 7Fh are done by a group of specific internal registers called Special Function Register (SFR) which may be addressed like RAM. SFRs may be bit addressable or byte addressable. The following table gives the SFRs and their addresses.

NAME	FUNCTION	INTERNAL RAM ADDRESS
A	Accumulator	0E0
B	Arithmetic	0F0
DPH	Addressing external memory	83
DPL	Addressing external memory	82
IE	Interrupt enable control	0A8
IP	Interrupt priority	0B8
P0	Input/output port latch	80
P1	Input/output port latch	90
P2	Input/output port latch	A0
P3	Input/output port latch	0B0
PCON	Power control	87
PSW	Program status word	0D0
SCON	Serial port control	98
SBUF	Serial port data buffer	99
SP	Stack pointer	81
TMOD	Timer/counter mode control	89
TCON	Timer counter control	88
TL0	Timer 0 low byte	8A
TH0	Timer 0 high byte	8C
TLI	Timer 1 low byte	8B
TH1	Timer 1 high byte	8D

**Note:

Any address used in the program must start with a number. Thus E0h for SFR begins with 0

COUNTERS AND TIMERS

The 8051 has two 16 – bit Timer/Counter registers – Timer 0 - T0 and Timer 1 – T1. The timers have three general functions

- Calculating time between events
- Counting events

- Generating baud rates for serial port.

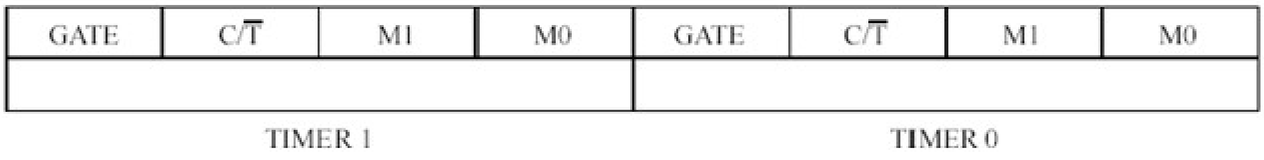
Since 8051 has 8 bit architecture, each 16 bit timer is accessed as two separate registers Timer low (TL0, TL1) and Timer high (TH0 , TH1). The word timer is used when it is used to count internal clock pulses and Counter is used when it is used to count external pulses. All Timer/Counter action is controlled by TMOD& TCON registers. .

When the microcontroller is programmed as timer, the timer increments by 1 for every machine cycle.The frequency of 8051 timers is - Internal clock frequency divided by 12.

E.g. If the oscillator frequency is 11.0592MHz, the timer frequency is 11.0592/12= 921.6 kHz
The time taken count from 0 to 50,000 is = 50000 x 1/ 921.6K Hz = 0.0542 s

BIT FORMAT OF TMOD REGISTER:

Timer/Counter Mode Control Register (Not Bit Addressable)



GATE	$C/\overline{T}$	M1	M0	GATE	$C/\overline{T}$	M1	M0
TIMER 1				TIMER 0			

GATE:

Every timer has a means of starting and stopping. This may be done either by hardware or by software controls. The hardware way of starting and stopping the timer by an external source is achieved by making GATE = 1. If the GATE = 0, the start and stop of the timer is done by software.

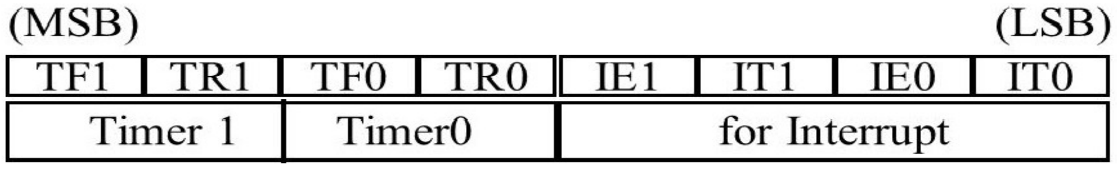
$C/\overline{T}$:

This bit is used to decide whether the timer is used as delay generator (timer) or as an event counter (counter). If C/T is = 0, it is used as timer. If C/T =1 it is used as counter.

M1,M0:
These two bits set the mode of operation of timers

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

BIT FORMAT OF TCON REGISTER:
The control of timer counter operation is done through the register called TCON. It is bit addressable. (This means that each bit can be read and altered individually). The bit format is as shown below.



(MSB)							(LSB)
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
Timer 1		Timer0		for Interrupt			

Bit	Symbol	TCON Bit Function
7	TF1	Timer 1 Overflow flag. Set when timer /counter over flows.
6	TR1	Timer 1 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to stop timer.
5	TF0	Timer 0 over flow flag, set when timer /counter overflows.
4	TR0	Timer 0 run control bit. Set to 1 by program to enable timer to count; cleared to 0 by program to halt timer.
3	IE1	External interrupt 1
2	IT1	External Timer interrupt 1 signal type control bit
1	IE0	External interrupt 0.
0	IT0	External timer interrupt 0 signal type control bit.

INTERRUPTS:

The interrupt is a process of data transfer by an external device that informs the CPU that it is ready for communication and requests attention.

8051 provided with 5 interrupts:

1. Timer 0 overflow interrupt – TF0
2. Timer 1 overflowinterrupt – TF1
3. External hardware interrupt INT0
4. External hardware interrupt INT1
5. Serial communication interrupts RI / TI.

The Timer and Serial interrupts are internally generated by the microcontroller, whereas the external interrupts are generated by additional interfacing devices or switches that are externally connected to the microcontroller. The interrupts are enabled or disabled by the Interrupt Enable (IE) register:

This register is responsible for enabling and disabling the interrupt. It is a bit addressable register in which EA must be set to one for enabling interrupts. In these five interrupts, if anyone interrupt is to be activated, the corresponding bit must be set. The bit format of IE register is as shown below.

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.

Interrupt priority:

It is desirable to set priorities among competing interrupts that may occur simultaneously. If two interrupts with same priority occur at the same time, then they have the following ranking:

1. IE0
2. TF0
3. IE1
4. TF1
5. Serial = RI OR TI

The priority levels of the interrupts can be changed by changing the corresponding bit in the Interrupt Priority (IP) register. The priority of interrupts can be set in the IP register. The bit format of IP register is as shown below.

D7 D0

--	--	PT2	PS	PT1	PX1	PT0	PX0

Priority bit = 1 assigns high priority. Priority bit = 0 assigns low priority.

--	IP.7	Reserved
--	IP.6	Reserved
PT2	IP.5	Timer 2 interrupt priority bit (8052 only)
PS	IP.4	Serial port interrupt priority bit
PT1	IP.3	Timer 1 interrupt priority bit
PX1	IP.2	External interrupt 1 priority bit
PT0	IP.1	Timer 0 interrupt priority bit
PX0	IP.0	External interrupt 0 priority bit

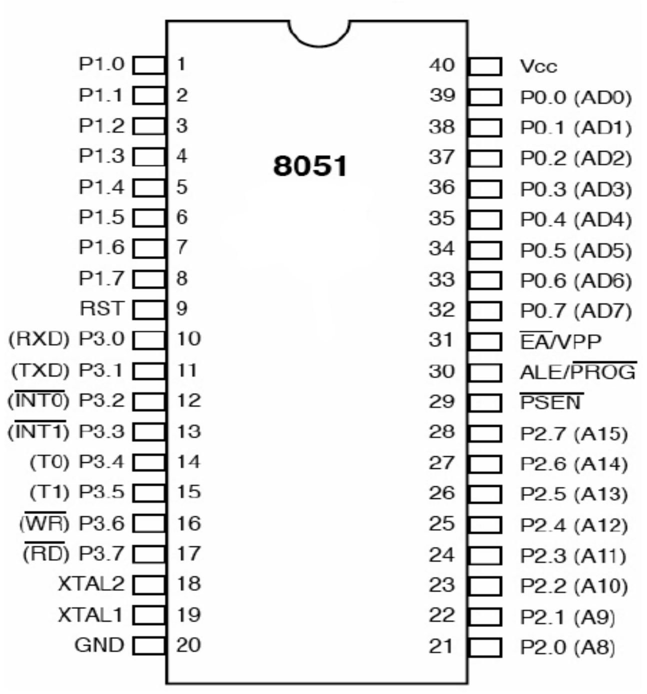
INTERRUPT DESTINATION

The addresslocations of various interrupts are as shown below. These are called vectored interrupts

Interrupt	Address (Hex)
IE0	0003
TF0	000B
IE1	0013
TF1	001B
SERIAL	0023

.

PIN OUT DIAGRAM OF 8051:



XTAL2 and XTAL 1 (Pin 18, 19)

XTAL2 and XTAL 1 pins are input output pins for the oscillator. These pins are used to connect an internal oscillator to the microcontroller.

Power supply connections (Pins 40, 20)

V_{cc}pin (Pin 40) is connected to +5V power supply and GND pin(Pin 20) is the circuit ground.

Port 0 (Pins 32- 39)

These pins can be used as input/ output pins when any external memory is not being used. When ALE or Pin 30 is at 1, then this port is used as data bus: When the ALE pin is at 0, then this port is used as a lower order address bus (A0 to A7)

Port 1 (Pins 1- 8)

Port 1 is an 8 bit bi-directional Input /Output port with internal pull up resistors. . Writing a 1 to the port latch causes it act as input.

Port 2 (Pins 21- 28)

Port 2 can be configured as Input / Output Pins. But, this is only possible when any external memory is not being used. If external memory is used, then these pins will work as higher order address bus (A8 to A15).

Port 3 (Pins 10- 17)

Port 3 pins can be used as universal Input or output pins. These pins have dual-function. The alternate functions of port 3 pins are as given below.

RST (Pin 9)

This is used for resetting the device.

ALE (Pin 30)

Address Latch Enable output is used for latching the lower order addresses byte during the external memory access.

PSEN (Pin 29)

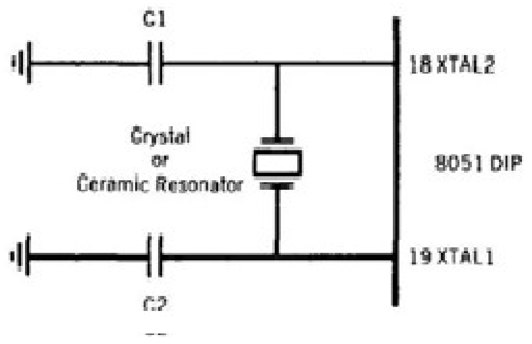
Program Store Enable is the output control signal. It is the read strobe to external program memory. During the internal program execution, it remains high.

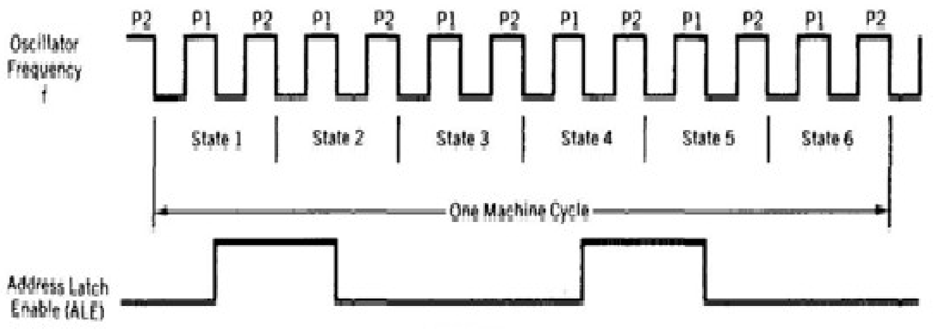
EA (Pin 31)

External Access pin, when held high executes instruction from the internal program memory till address 0FFFH; beyond this address, the instruction is fetched from external program memory. If this pin is low, all the instructions are fetched from the external memory.

The 8051 oscillator and clock

The heart of the 8051 is the circuitry that generates the clock pulses which are required for all internal operations. The two Capacitors and crystal oscillator are related to the clock circuitry which produces the system clock frequency. This crystal oscillator is used to generate clock pulses required for the synchronization of all the internal operations. For each instruction fetching, decoding, executing, and storing, controller requires clock pulse. (which is 1MHz to 16MHz).





One complete oscillation of the clock source is called a pulse. Two pulses forms a state and six states forms one machine cycle. Also note that, two pulses of ALE are available for 1 machine cycle.

Program instructions may require one or more machine cycles depending upon type of instructions. The time for execution of any instruction is calculated using the relation

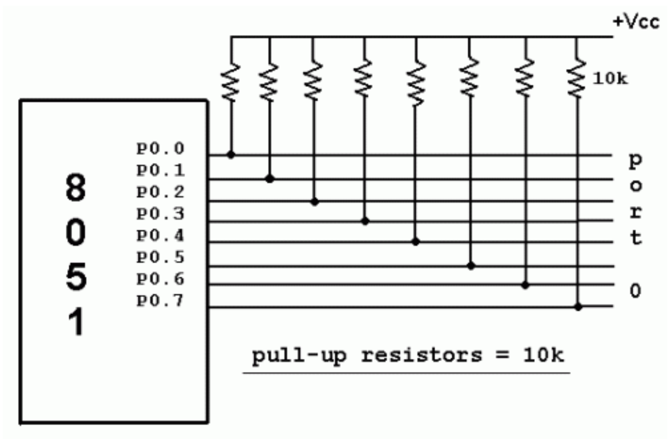
$$T = \frac{Cx\,12\,d}{crystalfrequency}$$

Where C is the number of Machine cycles.

I/O PORTS

A Port is a pin where data can be transferred between 8051 and an external device. 8051 has 4, 8 bit ports: P0, P1, P2 and P3. Apart from being used for input / output functions some of the pins have alternate functions.

PORT 0



This is an 8 bit, bit addressable input or output port. P0 has a dual role. Apart from being used as an I/O port, it is also used as a bidirectional lower order address and data bus for external memory.

To use P0 as input port, each pin must be connected to the 10 KΩ pull-up resistors externally as shown in the above diagram. To make it an input port, the port must be programmed by writing 1 to all bits. When 0 is written to the port it becomes an output.

PORT 1:

This is an 8 bit, bit addressable input or output port. In contrast to Port 0, this port does not need any pull-up resistors since it already has pull-up resistors internally. Upon reset, Port 1 is configured as an input port.

If Port 1 has been configured as an output port, to make it an input port again, it must be programmed as such by writing 1 to all its bits.

PORT 2:

This is an 8 bit, bit addressable input or output port. It has built-in pull-up resistors like in port 1. It must first be programmed by writing 1 to all bits required to be an input.

Dual role of port 2:

In many systems based on 8051, P2 is used as simple I/O. When 8051 is connected to external memory, P2 is used for the upper 8 bits of the external address and it cannot be used for I/O operations.

PORT 3:

Port 3 occupies a total of 8 pins. it can be used as input or output. Port 3 does not need pull-up resistors. Although port 3 is configured as an input port upon reset, this is not the way it is most commonly used. Port 3 has additional function of providing interrupt signals. The alternate functions of port 3 is listed below.

PORT 3 Pin	Function	Description
P3.0	RXD	Serial Input
P3.1	TXD	Serial Output
P3.2	INT0	External Interrupt 0
P3.3	INT1	External Interrupt 1
P3.4	T0	Timer 0
P3.5	T1	Timer 1
P3.6	WR	External Memory Write
P3.7	RD	External Memory Read