UNIT 1

8 BIT EMBEDDED PROCESSOR

1.1 INTRODUCTION

8051 is one of the first and most popular microcontrollers also known as MCS-51. Intel introduced it in the year 1981. Initially, it came out as an N-type metal-oxide-semiconductor (NMOS) based microcontroller, but later versions were based on complementary metal-oxide-semiconductor(CMOS) technology. These microcontrollers were named 80C51, where C in the name tells that it is based on CMOS technology. It is an 8-bit microcontroller which means the data bus is 8-bit. Therefore, it can process 8 bits at a time. It is used in a wide variety of embedded systems like robotics, remote controls, the automotive industry, telecom applications, power tools, etc.

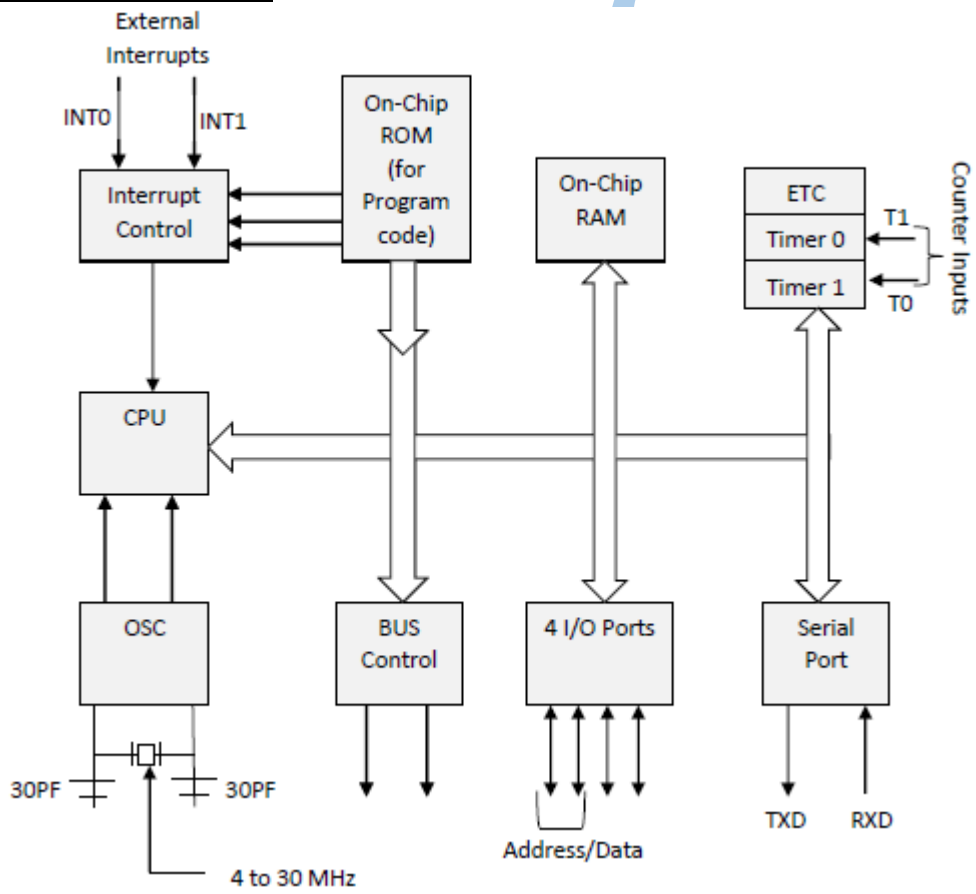
Microprocessor	Microcontroller
Microprocessors can be understood as the heart of a computer system.	Microcontrollers can be understood as the heart of an embedded system.
A microprocessor is a processor where the memory and I/O component are connected externally.	A microcontroller is a controlling device wherein the memory and I/O output component are present internally.
The circuit is complex due to external connection.	Microcontrollers are present on chip memory. The circuit is less complex.
The memory and I/O components are to be connected externally.	The memory and I/O components are available.
Microprocessors can't be used in compact system.	Microcontrollers can be used with a compact system.
Microprocessors are not efficient.	Microcontrollers are efficient.
Microprocessors have a zero status flag.	Microcontroller doesn't have a zero status flag.
Microprocessors have less number of registers.	Microcontrollers have more number of registers.

1.2 FEATURES OF 8051

- ✓ 4 KB on-chip ROM (Program memory).
- ✓ 128 bytes on-chip RAM (Data memory).
- ✓ The 8-bit data bus (bidirectional).

- ✓ 16-bit address bus (unidirectional).
- ✓ Two 16-bit timers.
- ✓ Instruction cycle of 1 microsecond with 12 MHz crystal.
- ✓ Four 8-bit input/output ports.
- ✓ 128 user-defined flags.
- ✓ Four register banks of 8 bit each.
- ✓ 16-byte bit-addressable RAM.
- ✓ The general-purpose registers are 32 each is 8-bit.
- ✓ 8051 has two external and three internal interrupts.
- ✓ 8051 microcontroller specifies some special function features like UARTs, ADC, Op-amp, etc.
- ✓ It has a 16-bit program counter and data pointer.

1.3 ARCHITECTURE OF 8051



Basic components present internally inside 8051 Microcontroller architecture are:

CPU (Central Processing Unit): CPU act as a mind of any processing machine. It synchronizes and manages all processes that are carried out in microcontroller. User has no power to control the functioning of CPU. It interprets the program stored in ROM and carries out from storage and then performs it projected duty. CPU manage the different types of registers available in 8051 microcontrollers.

Interrupts: Interrupts is a sub-routine call that given by the microcontroller when some other program with high priority is request for acquiring the system buses the n interrupts occur in current running program.

Interrupts provide a method to postpone or delay the current process, performs a sub-routine task and then restart the standard program again.

Types of interrupts in 8051 Microcontroller:

Let's see the five sources of interrupts in 8051 Microcontroller:

- Timer 0 overflow interrupt - TF0
- Timer 1 overflow interrupt - TF1
- External hardware interrupt - INT0
- External hardware interrupt - INT1
- Serial communication interrupt - RI/TI

Memory: For operation Micro-controller required a program. This program guides the microcontroller to perform the specific tasks. This program installed in microcontroller required some on chip memory for the storage of the program.

Microcontroller also required memory for storage of data and operands for the short duration. In microcontroller 8051 there is code or program memory of 4 KB that is it has 4 KB ROM and it also comprise of data memory (RAM) of 128 bytes.

Bus : Bus is a group of wires which uses as a communication canal or acts as means of data transfer. The different bus configuration includes 8, 16 or more cables. Therefore, a bus can bear 8 bits, 16 bits all together.

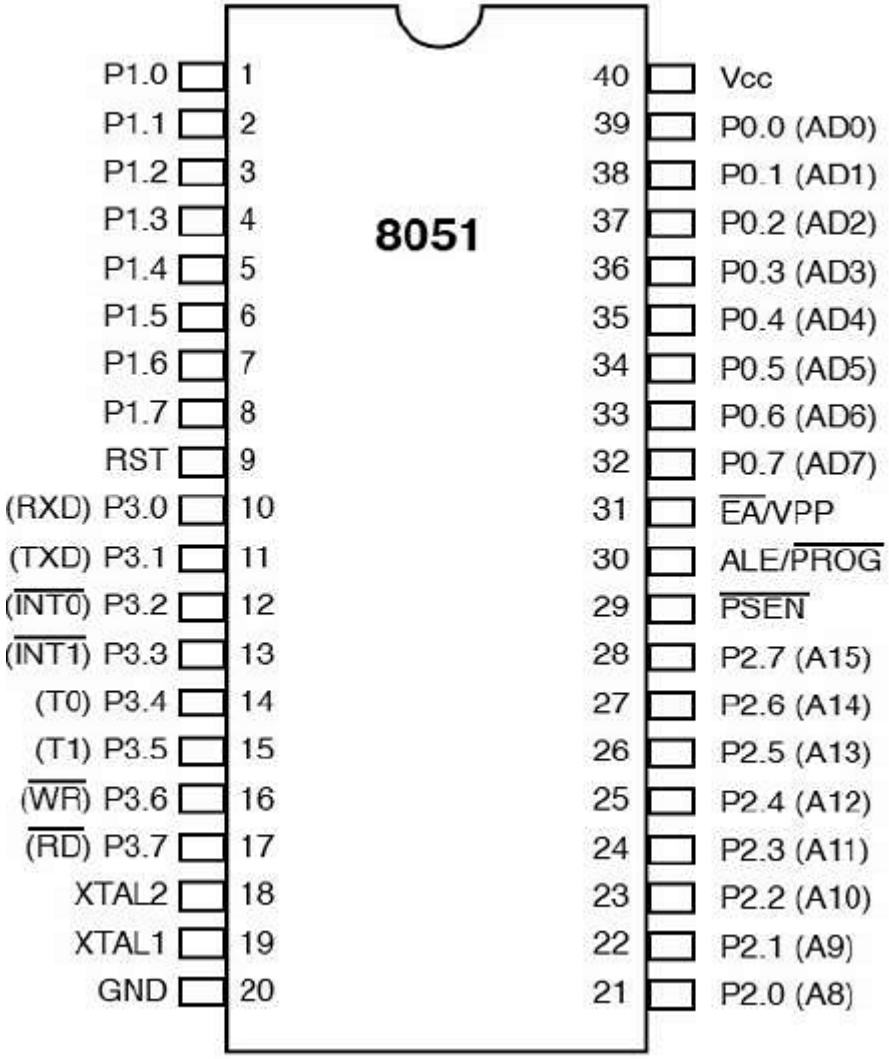
Types of buses in 8051 Microcontroller:

Let's see the two types of bus used in 8051 microcontroller:

- **Address Bus**: 8051 microcontrollers is consisting of 16 bit address bus. It is generally be used for transferring the data from Central Processing Unit to Memory.
- **Data bus**: 8051 microcontroller is consisting of 8 bits data bus. It is generally be used for transferring the data from one peripherals position to other peripherals.

Oscillator: As the microcontroller is digital circuit therefore it needs timer for their operation. To perform timer operation inside microcontroller it required externally connected or on-chip oscillator. Microcontroller is used inside an embedded system for managing the function of devices. Therefore, 8051 uses the two 16 bit counters and timers. For the operation of this timers and counters the oscillator is used inside microcontroller.

1.4 PIN DESCRIPTION OF 8051



- **Pins 1 to 8** − These pins are known as Port 1. This port doesn't serve any other functions. It is internally pulled up, bi-directional I/O port.
- **Pin 9** − It is a RESET pin, which is used to reset the microcontroller to its initial values.
- **Pins 10 to 17** − These pins are known as Port 3. This port serves some functions like interrupts, timer input, control signals, serial communication signals RxD and TxD, etc.
- **Pins 18 & 19** − These pins are used for interfacing an external crystal to get the system clock.
- **Pin 20** − This pin provides the power supply to the circuit.
- **Pins 21 to 28** − These pins are known as Port 2. It serves as I/O port. Higher order address bus signals are also multiplexed using this port.
- **Pin 29** − This is PSEN pin which stands for Program Store Enable. It is used to read a signal from the external program memory.
- **Pin 30** − This is EA pin which stands for External Access input. It is used to enable/disable the external memory interfacing.
- **Pin 31** − This is ALE pin which stands for Address Latch Enable. It is used to demultiplex the address-data signal of port.

- **Pins 32 to 39** − These pins are known as Port 0. It serves as I/O port. Lower order address and data bus signals are multiplexed using this port.
- **Pin 40** − This pin is used to provide power supply to the circuit.

8051 microcontrollers have 4 I/O ports each of 8-bit, which can be configured as input or output. Hence, total 32 input/output pins allow the microcontroller to be connected with the peripheral devices.

- **Pin configuration**, i.e. the pin can be configured as 1 for input and 0 for output as per the logic state.
  - **Input/Output (I/O) pin** − All the circuits within the microcontroller must be connected to one of its pins except P0 port because it does not have pull-up resistors built-in.
  - **Input pin** − Logic 1 is applied to a bit of the P register. The output FE transistor is turned off and the other pin remains connected to the power supply voltage over a pull-up resistor of high resistance.
- **Port 0** − The P0 (zero) port is characterized by two functions −
  - When the external memory is used then the lower address byte (addresses A0A7) is applied on it, else all bits of this port are configured as input/output.
  - When P0 port is configured as an output then other ports consisting of pins with built-in pull-up resistor connected by its end to 5V power supply, the pins of this port have this resistor left out.

Input Configuration

If any pin of this port is configured as an input, then it acts as if it "floats", i.e. the input has unlimited input resistance and in-determined potential.

Output Configuration

When the pin is configured as an output, then it acts as an "open drain". By applying logic 0 to a port bit, the appropriate pin will be connected to ground (0V), and applying logic 1, the external output will keep on "floating".

In order to apply logic 1 (5V) on this output pin, it is necessary to build an external pullup resistor.

Port 1

P1 is a true I/O port as it doesn't have any alternative functions as in P0, but this port can be configured as general I/O only. It has a built-in pull-up resistor and is completely compatible with TTL circuits.

Port 2

APEC

P2 is similar to P0 when the external memory is used. Pins of this port occupy addresses intended for the external memory chip. This port can be used for higher address byte with addresses A8-A15. When no memory is added then this port can be used as a general input/output port similar to Port 1.

Port 3

In this port, functions are similar to other ports except that the logic 1 must be applied to appropriate bit of the P3 register.

Pins Current Limitations

- When pins are configured as an output (i.e. logic 0), then the single port pins can receive a current of 10mA.
- When these pins are configured as inputs (i.e. logic 1), then built-in pull-up resistors provide very weak current, but can activate up to 4 TTL inputs of LS series.
- If all 8 bits of a port are active, then the total current must be limited to 15mA (port P0: 26mA).
- If all ports (32 bits) are active, then the total maximum current must be limited to 71mA.

Interrupts are the events that temporarily suspend the main program, pass the control to the external sources and execute their task. It then passes the control to the main program where it had left off.

8051 has 5 interrupt signals, i.e. INT0, TFO, INT1, TF1, RI/TI. Each interrupt can be enabled or disabled by setting bits of the IE register and the whole interrupt system can be disabled by clearing the EA bit of the same register.

IE (Interrupt Enable) Register

This register is responsible for enabling and disabling the interrupt. EA register is set to one for enabling interrupts and set to 0 for disabling the interrupts. Its bit sequence and their meanings are shown in the following figure.

EA	-	-	ES	ET1	EX1	ET0	EX0

EA	IE.7	It disables all interrupts. When EA = 0 no interrupt will be acknowledged and EA = 1 enables the interrupt individually.
-	IE.6	Reserved for future use.
-	IE.5	Reserved for future use.
ES	IE.4	Enables/disables serial port interrupt.

APEC

ET1	IE.3	Enables/disables timer1 overflow interrupt.
EX1	IE.2	Enables/disables external interrupt1.
ET0	IE.1	Enables/disables timer0 overflow interrupt.
EX0	IE.0	Enables/disables external interrupt0.

IP (Interrupt Priority) Register

We can change the priority levels of the interrupts by changing the corresponding bit in the Interrupt Priority (IP) register as shown in the following figure.

- A low priority interrupt can only be interrupted by the high priority interrupt, but not interrupted by another low priority interrupt.
- If two interrupts of different priority levels are received simultaneously, the request of higher priority level is served.
- If the requests of the same priority levels are received simultaneously, then the internal polling sequence determines which request is to be serviced.

-	-	PT2	PS	PT1	PX1	PT0	PX0
bit7	bit6	bit5	bit4	bit3	bit2	bit1	

-	IP.6	Reserved for future use.
-	IP.5	Reserved for future use.
PS	IP.4	It defines the serial port interrupt priority level.
PT1	IP.3	It defines the timer interrupt of 1 priority.
PX1	IP.2	It defines the external interrupt priority level.
PT0	IP.1	It defines the timer0 interrupt priority level.
PX0	IP.0	It defines the external interrupt of 0 priority level.

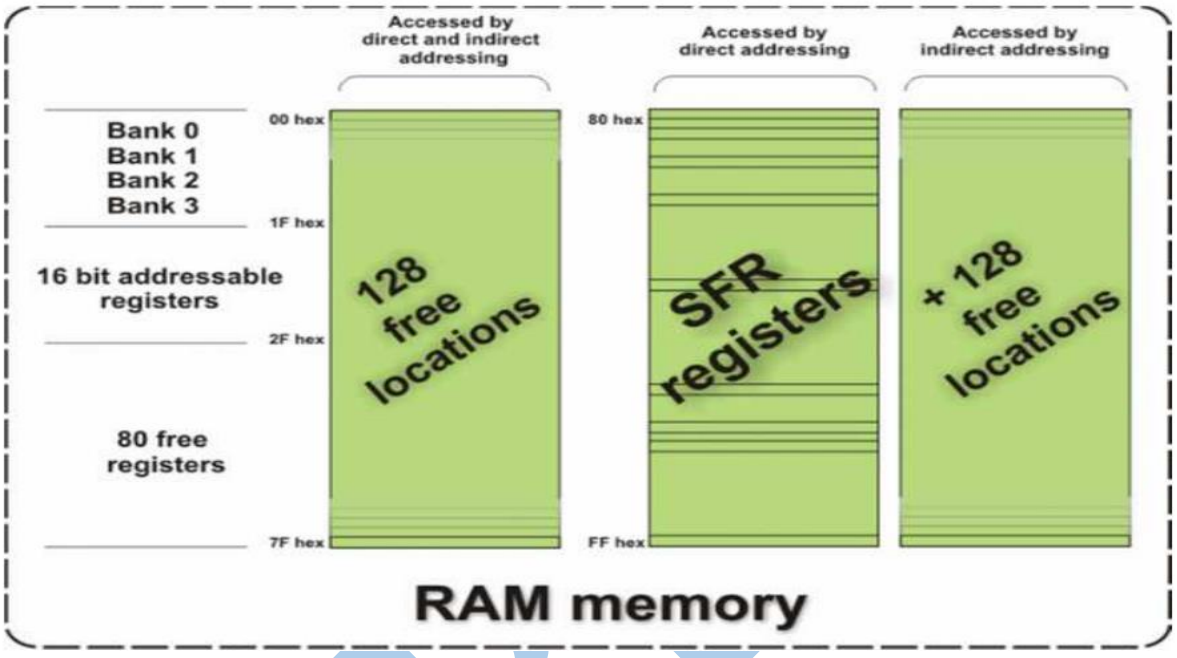
TCON Register

TCON register specifies the type of external interrupt to the microcontroller.

1.5 INTERNAL AND EXTERNAL MEMORIES

Internal Data Memory Up to 256 bytes of internal data memory are available depending on the 8051 derivative. Locations available to the user occupy addressing space from 0 to 7Fh, i.e. first 128 registers and this part of RAM is divided in several blocks. The first 128 bytes of internal data memory are both directly and indirectly addressable. The upper 128 bytes of data

memory (from 0x80 to 0xFF) can be addressed only indirectly. Since internal data memory is used for CALL stack also and there is only 256 bytes splited over few different memory areas fine utilizing of this memory is crucial for fast and compact code. Memory block in the range of 20h to 2Fh is bit-addressable, which means that each bit being there has its own address from 0 to 7Fh. Since there are 16 such registers, this block contains in total of 128 bits with separate addresses (Bit 0 of byte 20h has the bit address 0, and bit 7 of byte 2Fh has the bit address 7Fh).



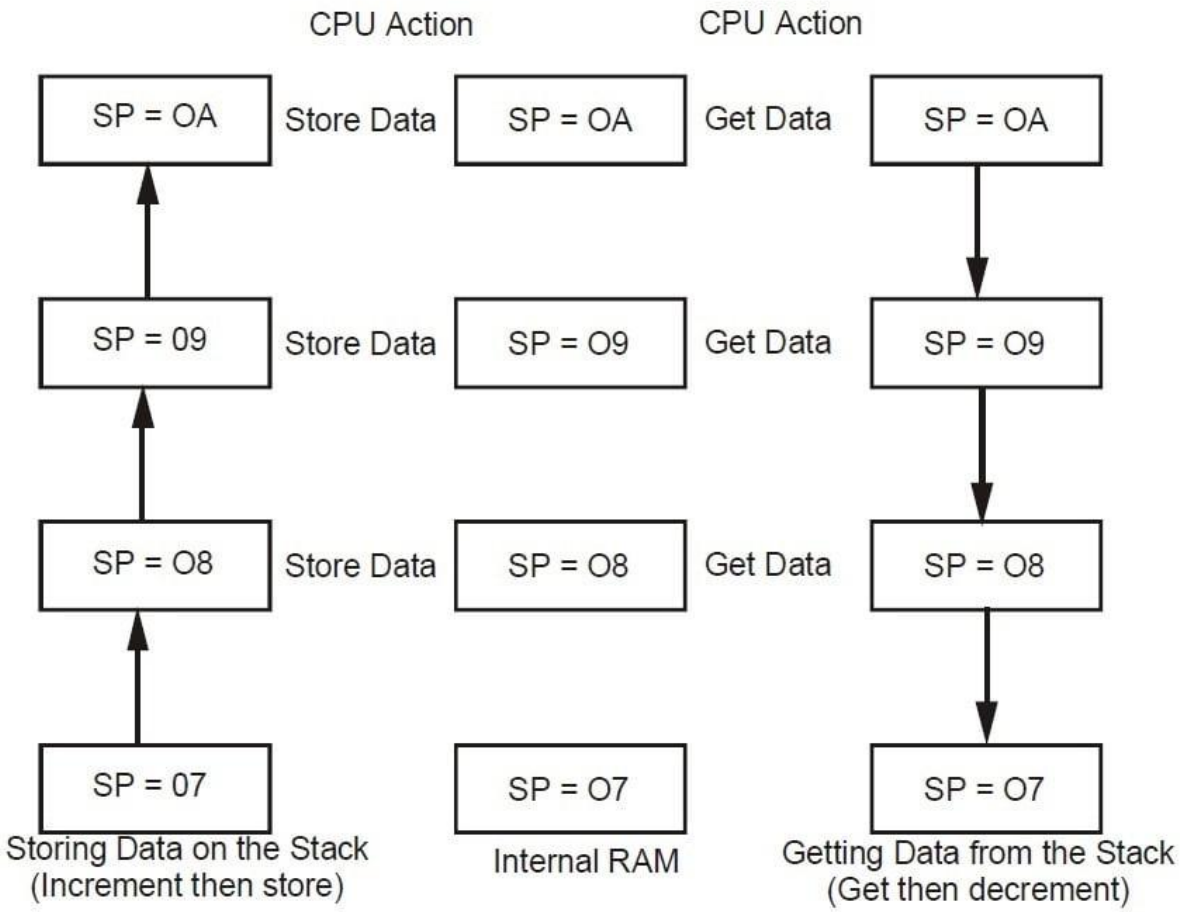
External Data Memory Access to external memory is slower than access to internal data memory. There may be up to 64K Bytes of external data memory. Several 8051 devices provide on-chip XRAM space that is accessed with the same instructions as the traditional external data space. This XRAM space is typically enabled via proper setting of SFR register and overlaps the external memory space. Setting of that register must be manually done in code, before any access to external memory or XRAM space is made.

1.6 STACK AND STACK POINTER

- The stack is a section of internal RAM used by the CPU to store information temporarily. This information could be data or an address.
- The CPU needs this storage area since there are only limited number of registers. The register to access the stack is called Stack Pointer.
- The stack pointer (SP) is an 8-bit register within the SFR area, with the address 81H and it can point an address location between 00h to FFh.
- Stack pointer is used to hold the address of Top of the Stack. The address held in the SP is the location in internal RAM where the last byte of data was stored by a stack operation.
- This register can hold one 8-bit address at a time, which is actually the memory location at top of the stack.
- The stack pointer is initialized to the value 07H when the 8051 microcontroller is reset.

Stack Operation

- When data is to be placed on the stack, the SP increments one before storing the data on the stack. When the data is retrieved from the stack, the SP decrements one after retrieving the data.
  Detailed stack operation shown in figure.



- In the 8051, the stack is configured as a series of memory locations following the Last-In First-Out (LIFO) pattern.
- In general, the stack is initialized in the internal RAM area. Any 8-bit data can be stored and retrieved from the stack using PUSH and POP instructions, with the help of the stack pointer.
- A push operation in the 8051 is used to store an 8-bit data in the stack.
- The PUSH instruction first increments the value of SP and then stores the data mentioned in the instruction in the memory location pointed to by SP.
- Similarly, the POP instruction stores the value from the top of the stack in the register mentioned in the instruction and then decrements the value of SP.
- The other instructions of the 8051 that affect the stack and the stack pointer are ACALL, LCALL, RET, and RETI.
- The stack pointer can be initialized to any internal RAM address by the programmer, by writing the required address in the SP SFR address 81H.

1.7 8051 ADDRESSING MODES

In 8051 there are 1-byte, 2-byte instructions and very few 3-byte instructions are present. The opcodes are 8-bit long. As the opcodes are 8-bit data, there are 256 possibilities. Among 256, 255 opcodes are implemented.

The clock frequency is12MHz, so 64 instruction types are executed in just 1 µs, and rest are just 2 µs. The Multiplication and Division operations take 4 µs to to execute.

In 8051 There are six types of addressing modes.

- Immediate Addressing Mode
- Register Addressing Mode
- Direct Addressing Mode
- Register Indirect Addressing Mode
- Indexed Addressing Mode
- Implied Addressing Mode

Immediate addressing mode

In this Immediate Addressing Mode, the data is provided in the instruction itself. The data is provided immediately after the opcode. These are some examples of Immediate Addressing Mode.

```
MOVA, #0AFH;
MOVR3, #45H;
MOVDPTR, #FE00H;
```

In these instructions, the # symbol is used for immediate data. In the last instruction, there is DPTR. The DPTR stands for Data Pointer. Using this, it points the external data memory location. In the first instruction, the immediate data is AFH, but one 0 is added at the beginning. So when the data is starting with A to F, the data should be preceded by 0.

Register addressing mode

In the register addressing mode the source or destination data should be present in a register (R0 to R7). These are some examples of RegisterAddressing Mode.

```
MOVA, R5;
MOVR2, #45H;
MOVR0, A;
```

In 8051, there is no instruction like **MOVR5, R7**. But we can get the same result by using this instruction **MOV R5, 07H**, or by using **MOV 05H, R7**. But this two instruction will work

when the selected register bank is **RB0**. To use another register bank and to get the same effect, we have to add the starting address of that register bank with the register number. For an example, if the RB2 is selected, and we want to access R5, then the address will be (10H + 05H = 15H), so the instruction will look like this **MOV 15H, R7**. Here 10H is the starting address of Register Bank 2.

Direct Addressing Mode

In the Direct Addressing Mode, the source or destination address is specified by using 8-bit data in the instruction. Only the internal data memory can be used in this mode. Here some of the examples of direct Addressing Mode.

```
MOV80H, R6;
MOVR2, 45H;
MOVR0, 05H;
```

The first instruction will send the content of registerR6 to port P0 (Address of Port 0 is 80H). The second one is forgetting content from 45H to R2. The third one is used to get data from Register R5 (When register bank RB0 is selected) to register R5.

Register indirect addressing Mode

In this mode, the source or destination address is given in the register. By using register indirect addressing mode, the internal or external addresses can be accessed. The R0 and R1 are used for 8-bit addresses, and DPTR is used for 16-bit addresses, no other registers can be used for addressing purposes. Let us see some examples of this mode.

```
MOV0E5H, @R0;
MOV@R1, 80H
```

In the instructions, the @ symbol is used for register indirect addressing. In the first instruction, it is showing that theR0 register is used. If the content of R0 is 40H, then that instruction will take the data which is located at location 40H of the internal RAM. In the second one, if the content of R1 is 30H, then it indicates that the content of port P0 will be stored at location 30H in the internal RAM.

```
MOVXA, @R1;
MOV@DPTR, A;
```

In these two instructions, the X in MOVX indicates the external data memory. The external data memory can only be accessed in register indirect mode. In the first instruction if the R0 is

holding 40H, then A will get the content of external RAM location40H. And in the second one, the content of A is overwritten in the location pointed by DPTR.

Indexed addressing mode

In the indexed addressing mode, the source memory can only be accessed from program memory only. The destination operand is always the register A. These are some examples of Indexed addressing mode.

```
MOVCA, @A+PC;
MOVCA, @A+DPTR;
```

The C in MOVC instruction refers to code byte. For the first instruction, let us consider A holds 30H. And the PC value is1125H. The contents of program memory location 1155H (30H + 1125H) are moved to register A.

Implied Addressing Mode

In the implied addressing mode, there will be a single operand. These types of instruction can work on specific registers only. These types of instructions are also known as register specific instruction. Here are some examples of Implied Addressing Mode.

```
RLA;
SWAPA;
```

These are 1- byte instruction. The first one is used to rotate the A register content to the Left. The second one is used to swap the nibbles in A.

1.8 CLASSIFICATION OF INSTRUCTION SET

We use data processing instructions to manipulate data within the registers.

Types for data processing instructions –

- Arithmetic instructions
- Logical instructions
- Multiply instructions
- Comparison instructions
- Move instructions

Most of the data processing instructions use a barrel shifter to pre-process the data of one of their operands. Each operation updates different flags in the cpsr (To learn more about 'CPSR' search "CPSR in ARM").

1.9 DATA TRANSFER INSTRUCTIONS

In 8051 Microcontroller there is 28 different instructions under the Data Transfer Group. In total there are 79 opcodes. The flags are not affected by using the data transfer

instructions, but the P (Parity) flag may change if the value of A register is changed using Data Transfer Instruction. Similarly, when a data is transferred to the PSW register, the flags will change.

In the following table, we will see the Mnemonics, Lengths, Execution Time in terms of the machine cycle, Number of Opcodes etc.

Mnemonics	Byte Count	Execution Time	Opcode Count
MOV A, Rn	1	1	8
MOV A, a8	2	1	1
MOV A, @Ri	1	1	2
MOV A, #d8	2	1	1
MOV Rn, A	1	1	8
MOV Rn, a8	2	2	8
MOV Rn, #d8	2	1	8
MOV a8, A	2	1	1
MOV a8, Rn	2	2	8
MOV a8, a8	3	2	1
MOV a8, @Ri	2	2	2
MOV a8, #d8	3	2	1
MOV @Ri, A	1	1	2
MOV @Ri, a8	2	2	2
MOV @Ri, #d8	2	1	2
MOV DPTR, #d16	3	2	1
MOVC A, @A+DPTR	1	2	1
MOVC A, @A+PC	1	2	1
MOVX A, @Ri	1	2	2
MOVX A, @DPTR	1	2	1

APEC

Mnemonics	Byte Count	Execution Time	Opcode Count
MOVX @Ri, A	1	2	2
MOVX @DPTR, A	1	2	1
PUSH a8	2	2	1
POP a8	2	2	1
XCH A, Rn	1	1	8
XCH A, a8	2	1	1
XCH A, @Ri	1	1	2
XCHD A, @Ri	1	1	2

If the Clock Frequency is 12MHz, then 1 machine cycle will take 1 μs for execution.

Examples

Sr.No	Instruction & Description
1	**MOV R2, 35H** It is an example of MOV Rn, a8. The content of location 35H of internal memory is moved to register R2.
2	**MOV 0F0H, #12H** It is an example of type MOV a8, #d8. Here the12H will be stored at register B. The equivalent instruction of this one is MOV B, #25H
3	**PUSH 0F0H** This is an example of PUSH a8. Here the contents of register B is pushed on the stack.
4	**POP 40H** This is like POP a8. It indicates the content of stack top is moved to the internal RAM location 40H.
5	**XCH A, R2** This is like XCH A, Rn. This instruction is used to exchange the content of A and register R2.
6	**XCHD A @R0** This instruction is like the instruction XCHD A,@Ri. Using this instruction the LS digit of A and LS digit of internal RAM location, which is pointed by R0 are swapped.

APEC

1.10 BYTE LEVEL LOGICAL INSTRUCTIONS

In 8051 Microcontroller there is 25 different instructions under the Logical Group. In total there are 49 opcodes. The Carry Flag (CY) affects only by instruction RRC and RLC.

In the following table, we will see the Mnemonics, Lengths, Execution Time in terms of the machine cycle, Number of Opcodes etc.

Mnemonics	Byte Count	ExecutionTime	Opcode Count
ANL A, Rn	1	1	8
ANL A, a8	2	1	1
ANL A, @Ri	1	1	2
ANL A, #d8	2	1	1
ANL a8, A	2	1	1
ANL a8, #d8	3	2	1
ORL A, Rn	1	1	8
ORL A, a8	2	1	1
ORL A, @Ri	1	1	2
ORL A, #d8	2	1	1
ORL a8, A	2	1	1
ORL a8, #d8	3	2	1
XRL A, Rn	1	1	8
XRL A, a8	2	1	1
XRL A, @Ri	1	1	2
XRL A, #d8	2	1	1
XRL a8, A	2	1	1

APEC

Mnemonics	Byte Count	ExecutionTime	Opcode Count
XRL a8, #d8	3	2	1
CLR A	1	1	1
CPL A	1	1	1
RL A	1	1	1
RLC A	1	1	1
RR A	1	1	1
RRC A	1	1	1
SWAP A	1	1	1

1.11 ARITHMETIC INSTRUCTIONS BIT LEVEL LOGICAL INSTRUCTIONS

In 8051 Microcontroller there are 24 different instructions under the Arithmetic Group. In total there are 64 opcodes. The Carry Flag (CY), Auxiliary Carry (AC)and Overflow flag (OV) are affected based on the result of ADD, ADDC, SUBB etc. instructions. The multiple and divide instructions clear the Carry flag, and also does not affect the AC flag. After execution of multiplication, the OV flag will be 1 when the result is greater than FFH. Otherwise, it is 0. Similarly, after division OV flag is 1 when the content of B is 00H before division, otherwise it is 0. The DA A (Decimal Adjust Accumulator) affects the Carry Flag only.

In the following table, we will see the Mnemonics, Lengths, Execution Time in terms of the machine cycle, Number of Opcodes etc.

Mnemonics	Byte Count	execution time	Opcode Count
ADD A, Rn	1	1	8
ADD A, a8	2	1	1
ADD A, @Ri	1	1	2

Mnemonics	Byte Count	execution time	Opcode Count
ADD A, #d8	2	1	1
ADDC A, Rn	1	1	8
ADDC A, a8	2	1	1
ADDC A, @Ri	1	1	2
ADDC A, #d8	2	1	1
SUBB A, Rn	1	1	8
SUBB A, a8	2	1	1
SUBB A, @Ri	1	1	2

1.12 ROTATE AND SWAP INSTRUCTIONS

The ability to rotate data is useful for inspecting bits of a byte without using individual bit opcodes. The A register can be rotated one bit position to the left or right with or without including the C flag in the rotation. If the C flag is not included, then the rotation involves the eight bits of the A register. If the C flag is included, then nine bits are involved in the rotation. Including the C flag enables the programmer to construct rotate operations involving any number of bytes.

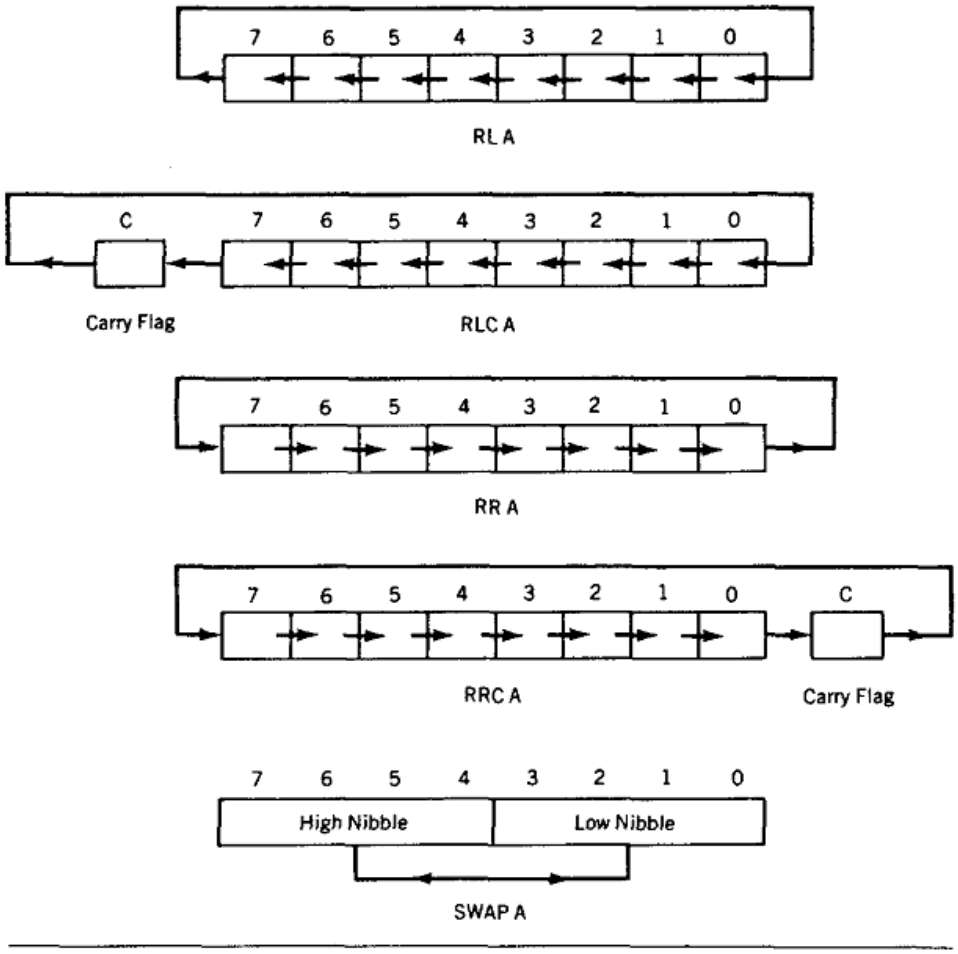
The SWAP instruction can be thought of as a rotation of nibbles in the A register.

Figure 4.2 diagrams the rotate and swap operations. which are given in the following table:

Mnemonic	Operation
RL A	Rotate the A register one bit position to the left; bit A0 to bit A I, A I to A2. A2 to A3. A3 to A4, A4 to A5, A5 to A6. A6 to A 7, and A 7 to A0
RLC A	Rotate the A register and the carry flag. as a ninth bit, one bit position to the left; bit A0 to bit Al, Al to A2, A2 to A3, A3 to A4, A4 to A5. A5 to A6, A6 to A 7, A 7 to the carry flag, and the carry flag to A0
RR A	Rotate the A register one bit position to the right; bit A0 to bit A 7, A6 to A5, A5 to A4, A4 to A3, A3 to A2. A2 to Al, and At to A0
RRC A	Rotate the A register and the carry flag, as a ninth bit, one bit position to the right; bit A0 to the carry flag. carry flag to A 7, A 7 to A6, A6 to A5, A5 to A4, A4 to A3, A3 to A2, A2 to Al, and Al to A0
SWAP A	Interchange the nibbles of register A; put the high nibble in the low nibble position and the low nibble in the high nibble position

Note that no flags. other than the carry flag in RRC and RLC, are affected. If the carry is used as part of a rotate instruction, the state of the carry flag should be known before the rotate is done.

FIGURE 4.2 *Register A Rotate Operations*



1.13 JUMP AND CALL INSTRUCTIONS

A **JUMP (JMP) instruction** is one in which the program control is transferred to a location in memory that is a part of the main program. It is utilized to cause the programmable logic control or PLC to skip over logics.

JUMP instruction uses the immediate addressing mode. Another important point about the JUMP instruction is that it does not require any return instruction after the execution of the JUMP instruction. Also, there is no need of initialization of a stack pointer.

Since the JUMP instruction is a transfer type instruction, we can transfer the information from one memory location to another using this instruction. The JUMP instruction requires only 3 machine cycles, which is less than that for the CALL instruction.

A **CALL instruction** is utilized to call a sub-routine. Using a CALL instruction, the program control is transferred to a location in memory that is not a part of the main program.

A CALL instruction necessarily requires the initialization of a Stack Pointer (SP). It is basically a control transfer type instruction because it is used to invoke the subroutine.

Once all the CALL instructions called the subroutine and the execution of these subroutine is completed, then the program control is transferred back to the caller by using the RET instruction.

If we talk about machine cycles for the CALL instruction, then it requires 5 machine cycles.

1.14 TIME DELAY FOR 8051

APEC