Microcontroller

A microcontroller is a compact integrated circuit designed to govern a specific operation in an embedded system. A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip.

Sometimes referred to as an embedded controller or microcontroller unit (MCU), microcontrollers are found in vehicles, robots, office machines, medical devices, mobile radio transceivers, vending machines and home appliances, among other devices. They are essentially simple miniature personal computers (PCs) designed to control small features of a larger component, without a complex front-end operating system (OS).

How do microcontrollers work?
A microcontroller is embedded inside of a system to control a singular function in a device. It does this by interpreting data it receives from its I/O peripherals using its central processor. The temporary information that the microcontroller receives is stored in its data memory, where the processor accesses it and uses instructions stored in its program memory to decipher and apply the incoming data. It then uses its I/O peripherals to communicate and enact the appropriate action.

Microcontrollers are used in a wide array of systems and devices. Devices often utilize multiple microcontrollers that work together within the device to handle their respective tasks.

For example, a car might have many microcontrollers that control various individual systems within, such as the anti-lock braking system, traction control, fuel injection or suspension control. All the microcontrollers communicate with each other to inform the correct actions. Some might communicate with a more complex central computer within the car, and others might only communicate with other microcontrollers. They send and receive data using their I/O peripherals and process that data to perform their designated tasks.

What are the elements of a microcontroller?
The core elements of a microcontroller are:

The processor (CPU) -- A processor can be thought of as the brain of the device. It processes and responds to various instructions that direct the microcontroller's function. This involves performing basic arithmetic, logic and I/O operations. It also performs data transfer operations, which communicate commands to other components in the larger embedded system.
Memory -- A microcontroller's memory is used to store the data that the processor receives and uses to respond to instructions that it's been programmed to carry out. A microcontroller has two main memory types:

Program memory, which stores long-term information about the instructions that the CPU carries out. Program memory is non-volatile memory, meaning it holds information over time without needing a power source.
Data memory, which is required for temporary data storage while the instructions are being executed. Data memory is volatile, meaning the data it holds is temporary and is only maintained if the device is connected to a power source.
I/O peripherals -- The input and output devices are the interface for the processor to the outside world. The input ports receive information and send it to the processor in the form of binary data. The processor receives that data and sends the necessary instructions to output devices that execute tasks external to the microcontroller.
While the processor, memory and I/O peripherals are the defining elements of the microprocessor, there are other elements that are frequently included. The term I/O peripherals itself simply refers to supporting components that interface with the memory and processor. There are many supporting components that can be classified as peripherals. Having some manifestation of an I/O peripheral is elemental to a microprocessor, because they are the mechanism through which the processor is applied.

Other supporting elements of a microcontroller include:

Analog to Digital Converter (ADC) -- An ADC is a circuit that converts analog signals to digital signals. It allows the processor at the center of the microcontroller to interface with external analog devices, such as sensors.
Digital to Analog Converter (DAC) -- A DAC performs the inverse function of an ADC and allows the processor at the center of the microcontroller to communicate its outgoing signals to external analog components.
System bus -- The system bus is the connective wire that links all components of the microcontroller together.
Serial port -- The serial port is one example of an I/O port that allows the microcontroller to connect to external components. It has a similar function to a USB or a parallel port but differs in the way it exchanges bits.
Microcontroller features
A microcontroller's processor will vary by application. Options range from the simple 4-bit, 8-bit or 16-bit processors to more complex 32-bit or 64-bit processors. Microcontrollers can use volatile memory types such as random access memory (RAM) and non-volatile memory types -- this includes flash memory, erasable programmable read-only memory (EPROM) and electrically erasable programmable read-only memory (EEPROM).

Generally, microcontrollers are designed to be readily usable without additional computing components because they are designed with sufficient onboard memory as well as offering pins for general I/O operations, so they can directly interface with sensors and other components.

Microcontroller architecture can be based on the Harvard architecture or von Neumann architecture, both offering different methods of exchanging data between the processor and memory. With a Harvard architecture, the data bus and instruction are separate, allowing for simultaneous transfers. With a Von Neumann architecture, one bus is used for both data and instructions.

Microcontroller processors can be based on complex instruction set computing (CISC) or reduced instruction set computing (RISC). CISC generally has around 80 instructions while RISC has about 30, as well as more addressing modes, 12-24 compared to RISC's 3-5. While CISC can be easier to implement and has more efficient memory use, it can have performance degradation due to the higher number of clock cycles needed to execute instructions. RISC, which places more emphasis on software, often provides better performance than CISC processors, which put more emphasis on hardware, due to its simplified instruction set and, therefore, increased design simplicity, but because of the emphasis it places on software, the software can be more complex. Which ISC is used varies depending on application.

When they first became available, microcontrollers solely used assembly language. Today, the C programming language is a popular option. Other common microprocessor languages include Python and JavaScript.

MCUs feature input and output pins to implement peripheral functions. Such functions include analog-to-digital converters, liquid crystal display (LCD) controllers, real-time clock (RTC), universal synchronous/asynchronous receiver transmitter (USART), timers, universal asynchronous receiver transmitter (UART) and universal serial bus (USB) connectivity. Sensors gathering data related to humidity and temperature, among others, are also often attached to microcontrollers.

Types of microcontrollers

Common MCUs include the Intel MCS-51, often referred to as an 8051 microcontroller, which was first developed in 1985; the AVR microcontroller developed by Atmel in 1996; the programmable interface controller (PIC) from Microchip Technology; and various licensed Advanced RISC Machines (ARM) microcontrollers.

A number of companies manufacture and sell microcontrollers, including NXP Semiconductors, Renesas Electronics, Silicon Labs and Texas Instruments.

Microcontroller applications

Microcontrollers are used in multiple industries and applications, including in the home and enterprise, building automation, manufacturing, robotics, automotive, lighting, smart energy, industrial automation, communications and internet of things (IoT) deployments.

One very specific application of a microcontroller is its use as a digital signal processor. Frequently, incoming analog signals come with a certain level of noise.

Noise in this context means ambiguous values that cannot be readily translated into standard digital values. A microcontroller can use its ADC and DAC to convert the incoming noisy analog signal into an even outgoing digital signal.

The simplest microcontrollers facilitate the operation of electromechanical systems found in everyday convenience items, such as ovens, refrigerators, toasters, mobile devices, key fobs, video game systems, televisions and lawn-watering systems. They are also common in office machines such as photocopiers, scanners, fax machines and printers, as well as smart meters, ATMs and security systems.

More sophisticated microcontrollers perform critical functions in aircraft, spacecraft, ocean-going vessels, vehicles, medical and life-support systems as well as in robots. In medical scenarios, microcontrollers can regulate the operations of an artificial heart, kidney or other organs. They can also be instrumental in the functioning of prosthetic devices.

Microcontrollers vs. microprocessors
The distinction between microcontrollers and microprocessors has gotten less clear as chip density and complexity has become relatively cheap to manufacture and microcontrollers have thus integrated more "general computer" types of functionality. On the whole, though, microcontrollers can be said to function usefully on their own, with a direct connection to sensors and actuators, where microprocessors are designed to maximize compute power on the chip, with internal bus connections (rather than direct I/O) to supporting hardware such as RAM and serial ports. Simply put, coffee makers use microcontrollers; desktop computers use microprocessors.

The Microchip Technology ATtiny817 microcontroller. MICROCHIP TECHNOLOGY INC.
The Microchip Technology ATtiny817 microcontroller.
Microcontrollers are less expensive and use less power than microprocessors. Microprocessors do not have built-in RAM, read-only memory (ROM) or other peripherals on the chip, but rather attach to these with their pins. A microprocessor can be considered the heart of a computer system, whereas a microcontroller can be considered the heart of an embedded system.

Choosing the right microcontroller
There are a number of technology and business considerations to keep in mind when choosing a microcontroller for a project. Beyond cost, it is important to consider the maximum speed, amount of RAM or ROM, number or types of I/O pins on an MCU, as well as power consumption and constraints and development support.

Embedded Processor

An embedded processor is a type of microprocessor designed into a system to control electrical and mechanical functions. Embedded processors are usually simple in design, limited in computational power and I/O capabilities, and have minimal power requirements. At a basic level, embedded processors are a CPU chip placed in a system that it helps control.

Embedded processors are often confused with microcontrollers. While they do perform similar functions, they integrate with their given system in different ways. The actual functions they perform can also be different as well.

Microcontrollers are the result of technological advances decreasing the size of controllers. Eventually, all of the components of a controller including I/O devices and memory evolved into a single chip, giving us the "micro" in microcontrollers. These chips are small, self-contained devices that have all of the features necessary to control the system they are embedded in.

This control autonomy is the primary difference between microcontrollers and embedded processors. Embedded processors require other external components such as integrated memory and peripheral interfaces to perform their designated functions. The two devices are frequently referred to as one device because embedded processors are often components within a microcontroller.

Introduction to 8051

The 8051 Microcontroller Introduction gives a brief overview about the 8051 Microcontroller and its history. Intel's 8051 Microcontroller (Intel MSC-51 Architecture) was a successor to 8048 Microcontroller (Intel MSC-48 Architecture).

Originally, 8051 Microcontrollers were developed using N-MOS Technology but the use of battery powered devices and their low power consumption lead to usage of CMOS Technology (which is famous for its low power consumption).

Even though Intel developed 8051 Microcontrollers (which is discontinued in 2007), more than 20 semiconductor manufacturers are still producing 8051 compatible microcontrollers i.e. processors based on MSC-51 Architecture.

Some of the 8051 Microcontrollers produced by different manufacturers are: Atmel (AT89C51, AT89S51), Phillips (S87C654), STC Micro (STC89C52), Infineon (SAB-C515, XC800), Siemens (SAB-C501), Silicon Labs (C8051), NXP (NXP700, NXP900), etc.

Majority of the modern 8051 Microcontrollers are Silicon IP Cores (Intellectual Property Cores) but discrete 8051 Microcontroller IC's are also available. Because of their low power consumption, smaller size and simple architecture, 8051 IP Cores are used in FPGAs (Field Programmable Gate Array) and SoCs (System on Chip) instead of Advanced ARM Architecture based MCUs.

Applications of 8051 Microcontroller

Even with the development of many advanced and superior Microcontrollers, 8051 Microcontroller is still being used in many embedded system and applications.

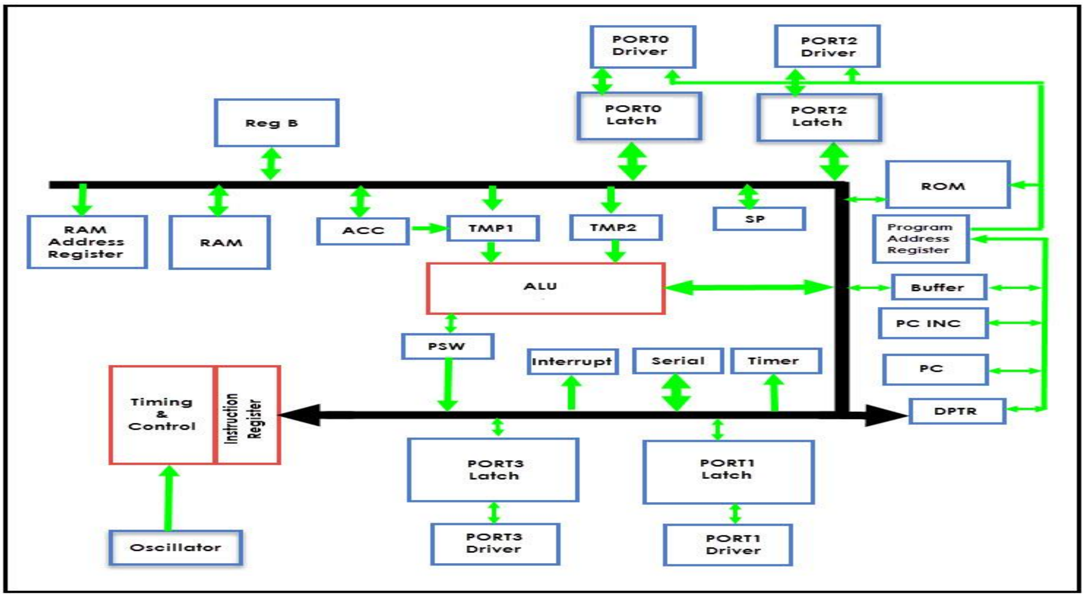
Some of the applications of 8051 Microcontroller are mentioned below:

Consumer Appliances (TV Tuners, Remote controls, Computers, Sewing Machines, etc.)
Home Applications (TVs, VCR, Video Games, Camcorder, Music Instruments, Home Security Systems, Garage Door Openers, etc.)
Communication Systems (Mobile Phones, Intercoms, Answering Machines, Paging Devices, etc.)
Office (Fax Machines, Printers, Copiers, Laser Printers, etc.)
Automobiles (Air Bags, ABS, Engine Control, Transmission Control, Temperature Control, Keyless Entry, etc)
Aeronautical and Space
Medical Equipment
Defense Systems
Robotics
Industrial Process and Flow Control
Radio and Networking Equipment
Remote Sensing

8051 Microcontroller Basics

8051 is an 8 – bit Microcontroller i.e. the data bus of the 8051 Microcontroller (both internal and external) is 8 – bit wide. It is a CISC based Microcontroller with Harvard Architecture (separate program and data memory).

Since the basic layout of a microcontroller includes a CPU, ROM, RAM, etc. the 8051 microcontrollers also has a similar layout. The following image shows a brief layout of a typical 8051 Microcontroller.



8051 Microcontroller Features

8 – Bit ALU: ALU or Arithmetic Logic Unit is the heart of a microcontroller. It performs arithmetic and bitwise operation on binary numbers. The ALU in 8051 is an 8 – Bit ALU i.e. it can perform operations on 8 – bit data.

8 – Bit Accumulator:The Accumulator is an important register associated with the ALU. The accumulator in 8051 is an 8 – bit register.

RAM: 8051 Microcontroller has 128 Bytes of RAM which includes SFRs and Input / Output Port Registers.

ROM: 8051 has 4 KB of on-chip ROM (Program Memory).

I/O Ports: 8051 has four 8 – bit Input / Output Ports which are bit addressable and bidirectional.

Timers / Counters: 8051 has two 16 – bit Timers / Counters.

Serial Port: 8051 supports full duplex UART Communication.

External Memory: 8051Microcontroller can access two 16 – bit address line at once: one each for RAM and ROM. The total external memory that an 8051 Microcontroller can access for RAM and ROM is 64KB (216 for each type).

Additional Features: Interrupts, on-chip oscillator, Boolean Processor, Power Down Mode, etc.

NOTE: Some of the features like size of RAM and ROM, number of Timers, etc. are not generic. They vary by manufacturer.

8051 Microcontroller Packaging

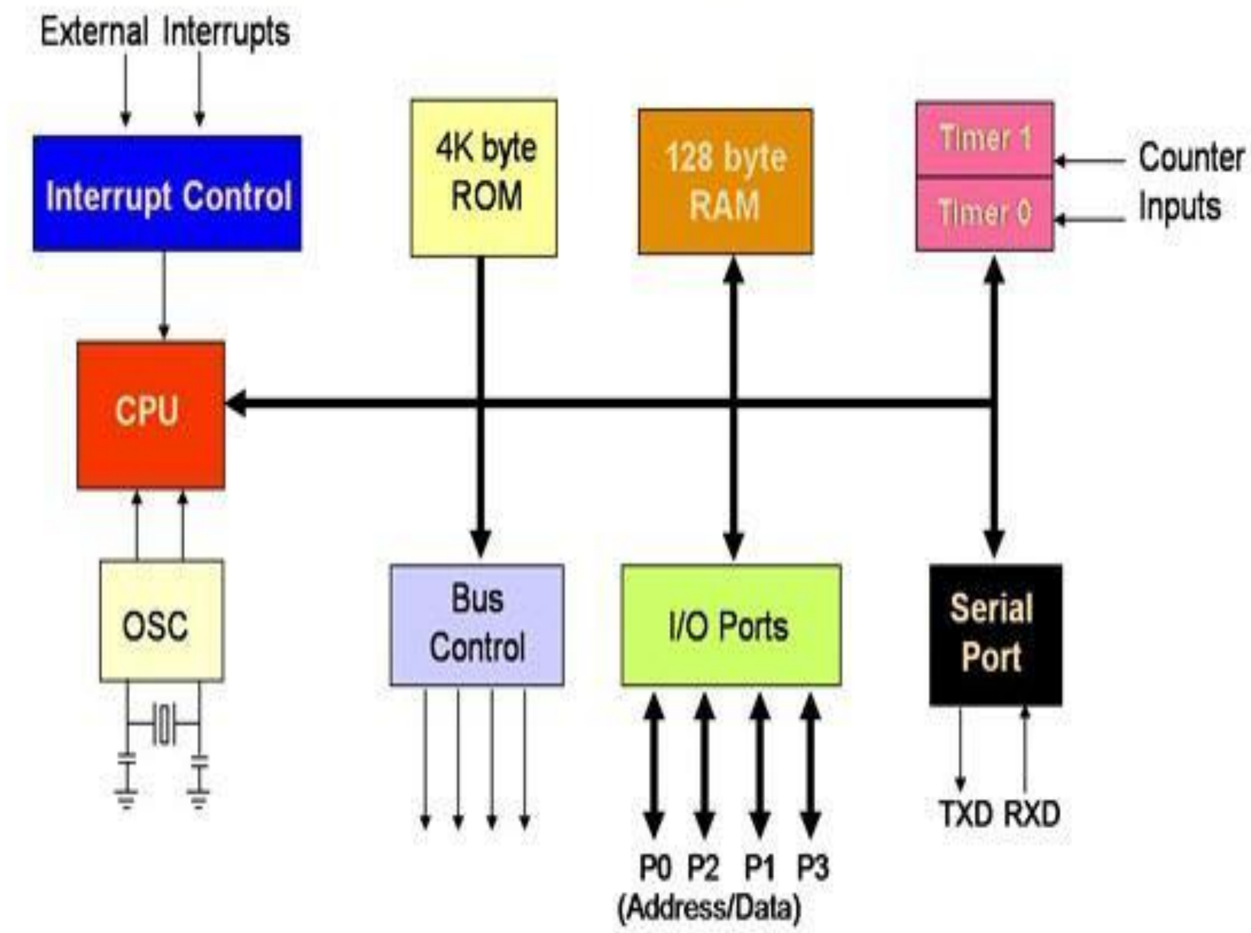
8051 Microcontroller is available in a variety of IC Packaging Types. The most popular and commonly used 8051 Microcontroller Packaging is Dual in-line or DIP. It is often available as a 40 – pin PDIP or Plastic DIP IC.

The other common packaging type is 44 – Lead PLCC (Plastic Leaded Chip Carrier). It is a kind of surface mount package.

Another surface mount packaging for 8051 microcontroller is 44 – Lead TQFP (Thin Quad Flat Package).

This article gave an introduction to 8051 Microcontroller and some its basic features. In the next article, we will see the Pinout Diagram, Pin Description and Architecture of 8051 Microcontroller.

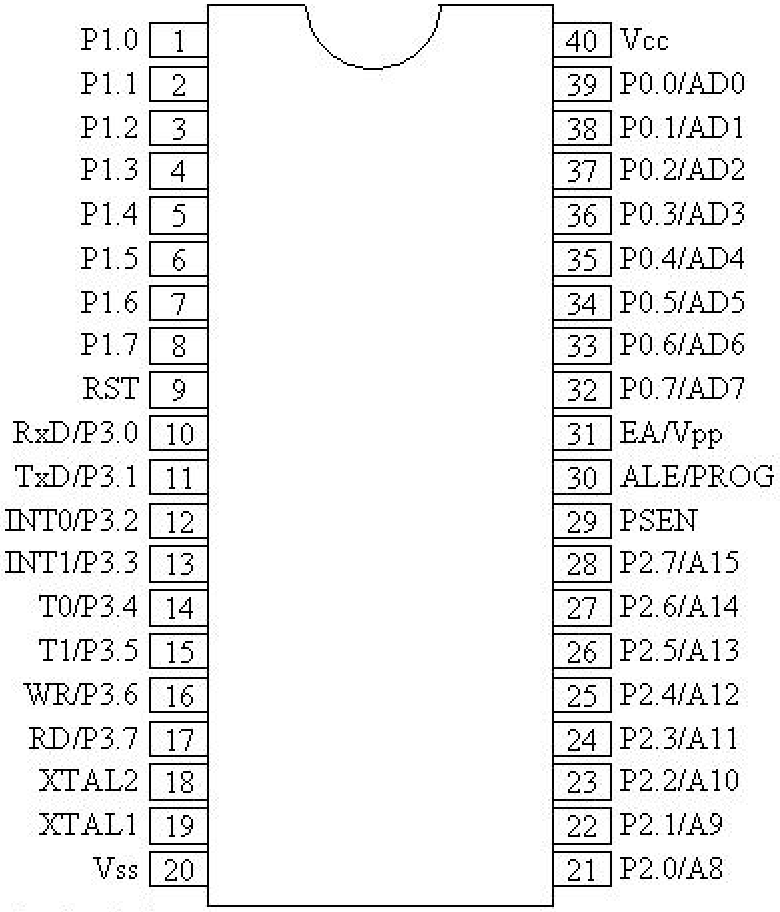
Block Diagram of 8051



Pins of 8051

In 8051, I/O operations are done using four ports and 40 pins. The following pin diagram shows the details of the 40 pins. I/O operation port reserves 32 pins where each port has 8 pins. The other 8 pins are designated as Vcc, GND, XTAL1, XTAL2, RST, EA (bar), ALE/PROG (bar), and PSEN (bar).

It is a 40 Pin PDIP (Plastic Dual Inline Package)



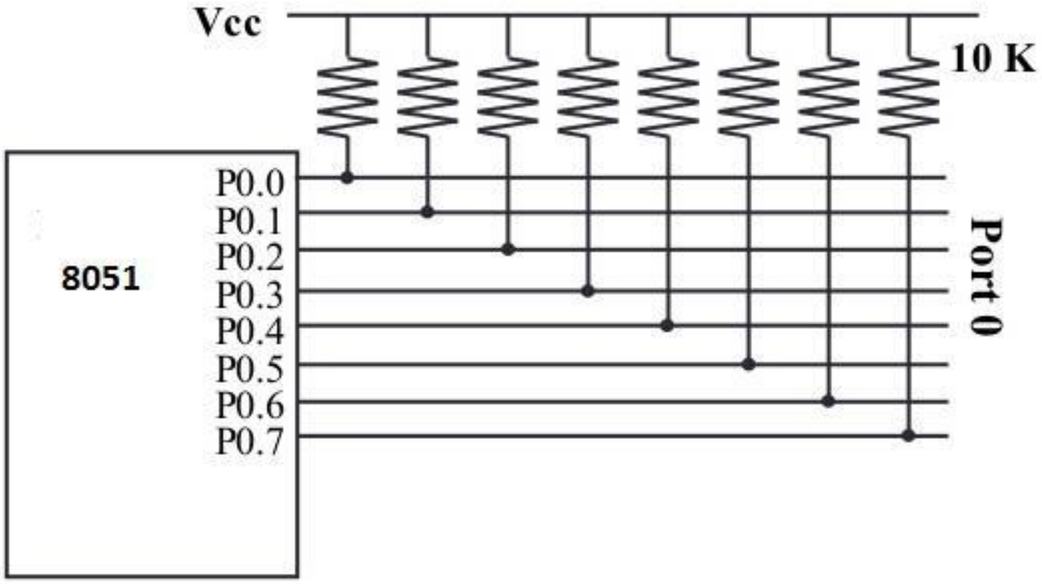
Note − In a DIP package, you can recognize the first pin and the last pin by the cut at the middle of the IC. The first pin is on the left of this cut mark and the last pin (i.e. the 40th pin in this case) is to the right of the cut mark.

I/O Ports and their Functions
The four ports P0, P1, P2, and P3, each use 8 pins, making them 8-bit ports. Upon RESET, all the ports are configured as inputs, ready to be used as input ports. When the first 0 is written to a port, it becomes an output. To reconfigure it as an input, a 1 must be sent to a port.

Port 0 (Pin No 32 – Pin No 39)
It has 8 pins (32 to 39). It can be used for input or output. Unlike P1, P2, and P3 ports, we normally connect P0 to 10K-ohm pull-up resistors to use it as an input or output port being an open drain.

It is also designated as AD0-AD7, allowing it to be used as both address and data. In case of 8031 (i.e. ROMless Chip), when we need to access the external ROM, then P0 will be used for both Address and Data Bus. ALE (Pin no 31) indicates if P0 has address or data. When ALE = 0, it provides data D0-D7, but when ALE = 1, it has address A0-A7. In case no external memory connection is available, P0 must be connected externally to a 10K-ohm pull-up resistor.

Port 0 diagram



```
MOV A,#0FFH  ;(comments: A=FFH(Hexadecimal  i.e. A=1111 1111)

MOV P0,A     ;(Port0 have 1's on every pin so that it works as Input)
```

Port 1 (Pin 1 through 8)

It is an 8-bit port (pin 1 through 8) and can be used either as input or output. It doesn't require pull-up resistors because they are already connected internally. Upon reset, Port 1 is configured as an input port. The following code can be used to send alternating values of 55H and AAH to Port 1.

```
;Toggle all bits of continuously
MOV     A,#55
BACK:

MOV     P2,A
ACALL   DELAY
CPL     A      ;complement(invert) reg. A
SJMP    BACK
```

If Port 1 is configured to be used as an output port, then to use it as an input port again, program it by writing 1 to all of its bits as in the following code.

```
;Toggle all bits of continuously

MOV     A ,#0FFH    ;A = FF hex
MOV     P1,A        ;Make P1 an input port
MOV     A,P1       ;get data from P1
MOV     R7,A       ;save it in Reg R7
ACALL   DELAY      ;wait

MOV     A,P1       ;get another data from P1
MOV     R6,A       ;save it in R6
ACALL   DELAY      ;wait

MOV     A,P1       ;get another data from P1
MOV     R5,A       ;save it in R5
```

Port 2 (Pins 21 through 28)

Port 2 occupies a total of 8 pins (pins 21 through 28) and can be used for both input and output operations. Just as P1 (Port 1), P2 also doesn't require external Pull-up resistors because they are already connected internally. It must be used along with P0 to provide the 16-bit address for the external memory. So it is also designated as (A0–A7), as shown in the pin diagram. When the 8051 is connected to an external memory, it provides path for upper 8-bits of 16-bits address, and it cannot be used as I/O. Upon reset, Port 2 is configured as an input port. The following code can be used to send alternating values of 55H and AAH to port 2.

```
;Toggle all bits of continuously
MOV     A,#55
BACK:
MOV     P2,A
ACALL   DELAY
CPL     A        ; complement(invert) reg. A
SJMP    BACK
```

If Port 2 is configured to be used as an output port, then to use it as an input port again, program it by writing 1 to all of its bits as in the following code.

```
;Get a byte from P2 and send it to P1
MOV    A,#0FFH    ;A = FF hex
MOV    P2,A       ;make P2 an input port
BACK:
MOV    A,P2       ;get data from P2
MOV    P1,A       ;send it to Port 1
SJMP   BACK       ;keep doing that
```

Port 3 (Pins 10 through 17)

It is also of 8 bits and can be used as Input/Output. This port provides some extremely important signals. P3.0 and P3.1 are RxD (Receiver) and TxD (Transmitter) respectively and are collectively used for Serial Communication. P3.2 and P3.3 pins are used for external interrupts. P3.4 and P3.5 are used for timers T0 and T1 respectively. P3.6 and P3.7 are Write (WR) and Read (RD) pins. These are active low pins, means they will be active when 0 is given to them and these are used to provide Read and Write operations to External ROM in 8031 based systems.

P3 Bit	Function	Pin
P3.0	RxD	10
P3.1 <	TxD	11
P3.2 <	Complement of INT0	12
P3.3 <	INT1	13
P3.4 <	T0	14
P3.5 <	T1	15
P3.6 <	WR	16
P3.7 <	Complement of RD	17

Dual Role of Port 0 and Port 2

Dual role of Port 0 − Port 0 is also designated as AD0–AD7, as it can be used for both data and address handling. While connecting an 8051 to external memory, Port 0 can provide both address and data. The 8051 microcontroller then multiplexes the input as address or data in order to save pins.

Dual role of Port 2 − Besides working as I/O, Port P2 is also used to provide 16-bit address bus for external memory along with Port 0. Port P2 is also designated as (A8–A15), while Port 0 provides the lower 8-bits via A0–A7. In other words, we can say that when an 8051 is connected to an external memory (ROM) which can be maximum up to 64KB and this is possible by 16 bit address bus because we know 216 = 64KB. Port2 is used for the upper 8-bit of the 16 bits address, and it cannot be used for I/O and this is the way any Program code of external ROM is addressed.

Hardware Connection of Pins

Vcc − Pin 40 provides supply to the Chip and it is +5 V.

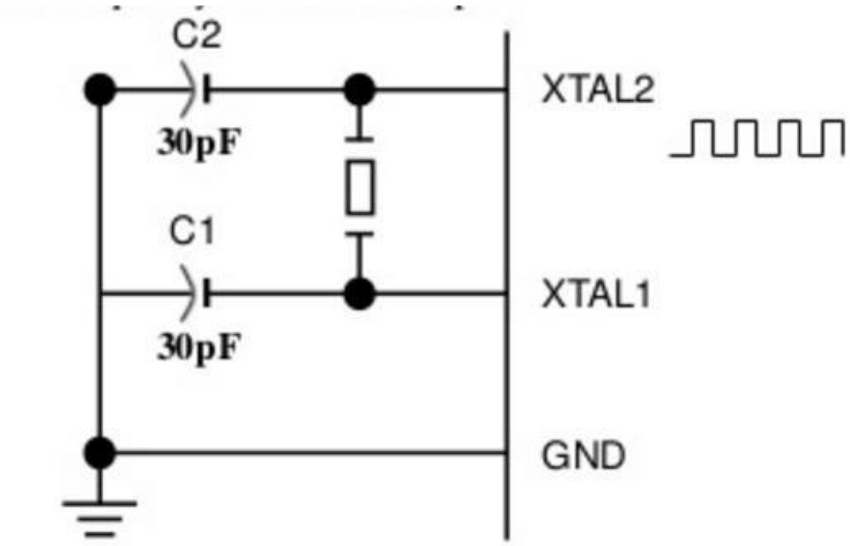
Gnd − Pin 20 provides ground for the Reference.

XTAL1, XTAL2 (Pin no 18 & Pin no 19) − 8051 has on-chip oscillator but requires external clock to run it. A quartz crystal is connected between the XTAL1 & XTAL2 pin of the chip. This crystal also needs two capacitors of 30pF for generating a signal of desired frequency. One side of each capacitor is connected to ground. 8051 IC is available in various speeds and it all depends on this Quartz crystal, for example, a 20 MHz microcontroller requires a crystal with a frequency no more than 20 MHz.

XTAL1, XTAL2 diagram

RST (Pin No. 9) − It is an Input pin and active High pin. Upon applying a high pulse on this pin, that is 1, the microcontroller will reset and terminate all activities. This process is known as Power-On Reset. Activating a power-on reset will cause all values in the register to be lost. It will set a program counter to all 0's. To ensure a valid input of Reset, the high pulse must be high for a minimum of two machine cycles before it is allowed to go low, which depends on the capacitor value and the rate at which it charges. (Machine Cycle is the minimum amount of frequency a single instruction requires in execution).

EA or External Access (Pin No. 31) − It is an input pin. This pin is an active low pin; upon applying a low pulse, it gets activated. In case of microcontroller (8051/52) having on-chip ROM, the EA (bar) pin is connected to Vcc. But in an 8031 microcontroller which does not have an on-chip ROM, the code is stored in an external ROM and then fetched by the microcontroller. In this case, we must connect the (pin no 31) EA to Gnd to indicate that the program code is stored externally.



RST, EA diagram

PSEN or Program store Enable (Pin No 29) − This is also an active low pin, i.e., it gets activated after applying a low pulse. It is an output pin and used along with the EA pin in 8031 based (i.e. ROMLESS) Systems to allow storage of program code in external ROM.

ALE or (Address Latch Enable) − This is an Output Pin and is active high. It is especially used for 8031 IC to connect it to the external memory. It can be used while deciding whether P0 pins will be used as Address bus or Data bus. When ALE = 1, then the P0 pins work as Data bus and when ALE = 0, then the P0 pins act as Address bus.

I/O Ports and Bit Addressability

It is a most widely used feature of 8051 while writing code for 8051. Sometimes we need to access only 1 or 2 bits of the port instead of the entire 8-bits. 8051 provides the capability to access individual bits of the ports.

While accessing a port in a single-bit manner, we use the syntax "SETB X. Y" where X is the port number (0 to 3), and Y is a bit number (0 to 7) for data bits D0-D7 where D0 is the LSB and D7 is the MSB. For example, "SETB P1.5" sets high bit 5 of port 1.

The following code shows how we can toggle the bit P1.2 continuously.

```
AGAIN:
SETB    P1.2
ACALL   DELAY
CLR     P1.2
ACALL   DELAY
SJMP    AGAIN
```

Single-Bit Instructions

Instructions	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)

Registers

Registers are used in the CPU to store information on temporarily basis which could be data to be processed, or an address pointing to the data which is to be fetched. In 8051, there is one data type is of 8-bits, from the MSB (most significant bit) D7 to the LSB (least significant bit) D0. With 8-bit data type, any data type larger than 8-bits must be broken into 8-bit chunks before it is processed.

The most widely used registers of the 8051 are A (accumulator), B, R0-R7, DPTR (data pointer), and PC (program counter). All these registers are of 8-bits, except DPTR and PC.

Storage Registers in 8051
We will discuss the following types of storage registers here −

Accumulator
R register
B register
Data Pointer (DPTR)
Program Counter (PC)
Stack Pointer (SP)

Accumulator

The accumulator, register A, is used for all arithmetic and logic operations. If the accumulator is not present, then every result of each calculation (addition, multiplication, shift, etc.) is to be stored into the main memory. Access to main memory is slower than access to a register like the accumulator because the technology used for the large main memory is slower (but cheaper) than that used for a register.

The "R" Registers

The "R" registers are a set of eight registers, namely, R0, R1 to R7. These registers function as auxiliary or temporary storage registers in many operations. Consider an example of the sum of 10 and 20. Store a variable 10 in an accumulator and another variable 20 in, say, register R4. To process the addition operation, execute the following command −

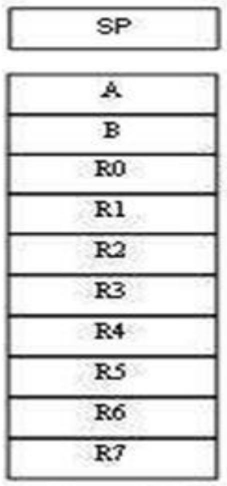
ADD A,R4

After executing this instruction, the accumulator will contain the value 30. Thus "R" registers are very important auxiliary or helper registers. The Accumulator alone would not be very useful if it were not for these "R" registers. The "R" registers are meant for temporarily storage of values.

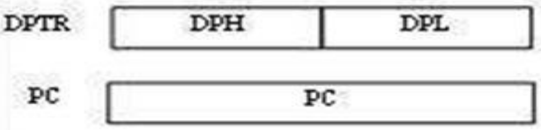
Let us take another example. We will add the values in R1 and R2 together and then subtract the values of R3 and R4 from the result.

```
MOV A,R3   ;Move the value of R3 into the accumulator
ADD A,R4   ;Add the value of R4
MOV R5,A   ;Store the resulting value temporarily in R5
MOV A,R1   ;Move the value of R1 into the accumulator
ADD A,R2   ;Add the value of R2
SUBB A,R5  ;Subtract the value of R5 (which now contains R3 + R4)
```

As you can see, we used R5 to temporarily hold the sum of R3 and R4. Of course, this is not the most efficient way to calculate (R1 + R2) – (R3 + R4), but it does illustrate the use of the "R" registers as a way to store values temporarily.



8-bit Registers of 8051



16-bit Registers of 8051

8 Bit registers

The "B" Register

The "B" register is very similar to the Accumulator in the sense that it may hold an 8-bit (1-byte) value. The "B" register is used only by two 8051 instructions: MUL AB and DIV AB. To quickly and easily multiply or divide A by another number, you may store the other number in "B" and make use of these two instructions. Apart from using MUL and DIV instructions, the "B" register is often used as yet another temporary storage register, much like a ninth R register.

The Data Pointer

The Data Pointer (DPTR) is the 8051's only user-accessible 16-bit (2-byte) register. The Accumulator, R0–R7 registers and B register are 1-byte value registers. DPTR is meant for pointing to data. It is used by the 8051 to access external memory using the address indicated by DPTR. DPTR is the only 16-bit register available and is often used to store 2-byte values.

The Program Counter

The Program Counter (PC) is a 2-byte address which tells the 8051 where the next instruction to execute can be found in the memory. PC starts at 0000h when the 8051 initializes and is incremented every time after an instruction is executed. PC is not always incremented by 1. Some instructions may require 2 or 3 bytes; in such cases, the PC will be incremented by 2 or 3.

Branch, jump, and interrupt operations load the Program Counter with an address other than the next sequential location. Activating a power-on reset will cause all values in the register to be lost. It means the value of the PC is 0 upon reset, forcing the CPU to fetch the first opcode from the ROM location 0000. It means we must place the first byte of upcode in ROM location 0000 because that is where the CPU expects to find the first instruction.

The Stack Pointer (SP)

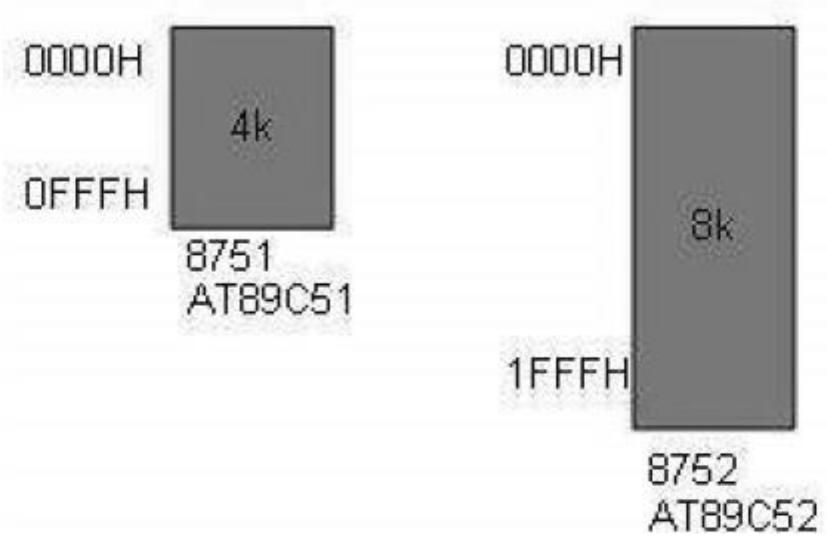
The Stack Pointer, like all registers except DPTR and PC, may hold an 8-bit (1-byte) value. The Stack Pointer tells the location from where the next value is to be removed from the stack. When a value is pushed onto the stack, the value of SP is incremented and then the value is stored at the resulting memory location. When a value is popped off the stack, the value is returned from the memory location indicated by SP, and then the value of SP is decremented.

This order of operation is important. SP will be initialized to 07h when the 8051 is initialized. If a value is pushed onto the stack at the same time, the value will be stored in the internal RAM address 08h because the 8051 will first increment the value of SP (from 07h to 08h) and then will store the pushed value at that memory address (08h). SP is modified directly by the 8051 by six instructions: PUSH, POP, ACALL, LCALL, RET, and RETI.

ROM Space in 8051

Some family members of 8051 have only 4K bytes of on-chip ROM (e.g. 8751, AT8951); some have 8K ROM like AT89C52, and there are some family members with 32K bytes and 64K bytes of on-chip ROM such as Dallas Semiconductor. The point to remember is that no member of the 8051 family can access more than 64K bytes of opcode since the program counter in 8051 is a 16-bit register (0000 to FFFF address).

The first location of the program ROM inside the 8051 has the address of 0000H, whereas the last location can be different depending on the size of the ROM on the chip. Among the 8051 family members, AT8951 has $k bytes of on-chip ROM having a memory address of 0000 (first location) to 0FFFH (last location).



ROM Space

8051 Flag Bits and PSW Register

The program status word (PSW) register is an 8-bit register, also known as flag register. It is of 8-bit wide but only 6-bit of it is used. The two unused bits are user-defined flags. Four of the flags are called conditional flags, which means that they indicate a condition which results after an instruction is executed. These four are CY

(Carry), AC (auxiliary carry), P (parity), and OV (overflow). The bits RS0 and RS1 are used to change the bank registers. The following figure shows the program status word register.

The PSW Register contains that status bits that reflect the current status of the CPU.

CY	CA	F0	RS1	RS0	OV	-	P

CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry Flag
F0	PSW.5	Flag 0 available to 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 FLAG
P	PSW.0	Parity FLAG. Set/ cleared by hardware during instruction cycle to indicate even/odd number of 1 bit in accumulator.

We can select the corresponding Register Bank bit using RS0 and RS1 bits.

RS1	RS2	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

CY, the carry flag − This carry flag is set (1) whenever there is a carry out from the D7 bit. It is affected after an 8-bit addition or subtraction operation. It can also be reset to 1 or 0 directly by an instruction such as "SETB C" and "CLR C" where "SETB" stands for set bit carry and "CLR" stands for clear carry.

AC, auxiliary carry flag − If there is a carry from D3 and D4 during an ADD or SUB operation, the AC bit is set; otherwise, it is cleared. It is used for the instruction to perform binary coded decimal arithmetic.

P, the parity flag − The parity flag represents the number of 1's in the accumulator register only. If the A register contains odd number of 1's, then P = 1; and for even number of 1's, P = 0.

OV, the overflow flag − This flag is set whenever the result of a signed number operation is too large causing the high-order bit to overflow into the sign bit. It is used only to detect errors in signed arithmetic operations.

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

MOV A, #9CH

ADD A, # 64H

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

CY = 1 since there is a carry beyond D7 bit
AC = 0 since there is a carry from D3 to D4
P = 0 because the accumulator has even number of 1's

Register Bank and STACK

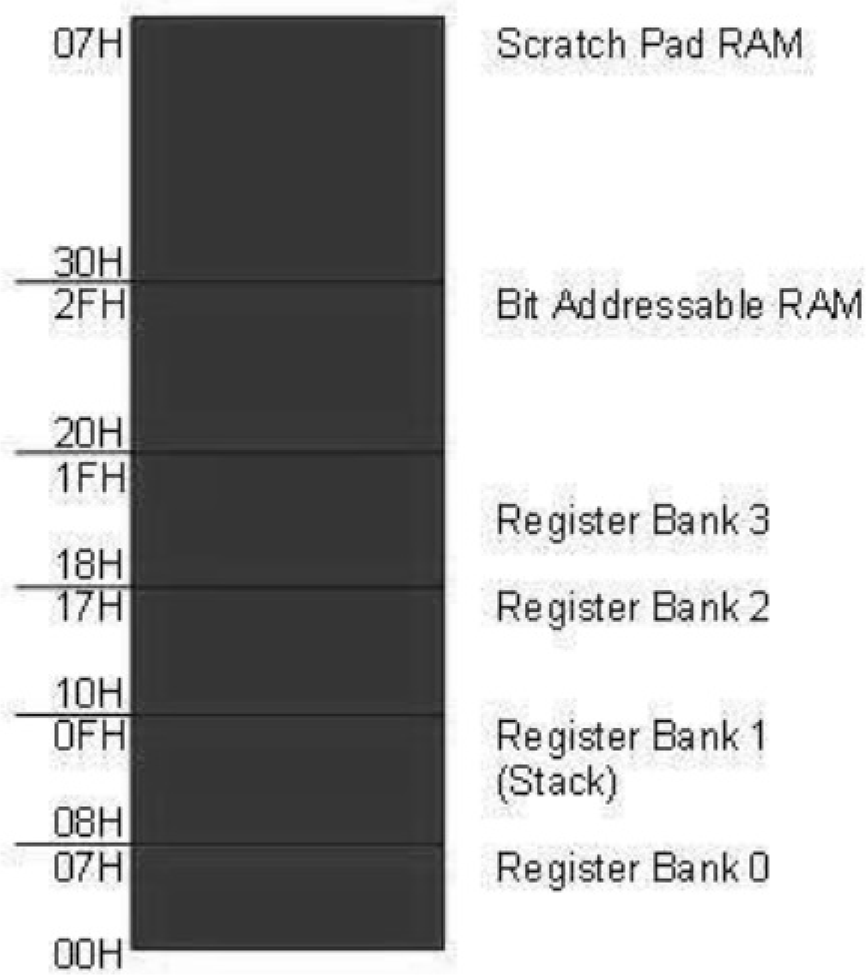
The 8051 microcontroller has a total of 128 bytes of RAM. We will discuss about the allocation of these 128 bytes of RAM and examine their usage as stack and register.

RAM Memory Space Allocation in 8051
The 128 bytes of RAM inside the 8051 are assigned the address 00 to 7FH. They can be accessed directly as memory locations and are divided into three different groups as follows −

32 bytes from 00H to 1FH locations are set aside for register banks and the stack.

16 bytes from 20H to 2FH locations are set aside for bit-addressable read/write memory.

80 bytes from 30H to 7FH locations are used for read and write storage; it is called as scratch pad. These 80 locations RAM are widely used for the purpose of storing data and parameters by 8051 programmers.



ROM Space Allocation

Register Banks in 8051

A total of 32 bytes of RAM are set aside for the register banks and the stack. These 32 bytes are divided into four register banks in which each bank has 8 registers, R0–R7. RAM locations from 0 to 7 are set aside for bank 0 of R0–R7 where R0 is RAM location 0, R1 is RAM location 1, R2 is location 2, and so on, until the memory location 7, which belongs to R7 of bank 0.

The second bank of registers R0–R7 starts at RAM location 08 and goes to locations OFH. The third bank of R0–R7 starts at memory location 10H and goes to location to 17H. Finally, RAM locations 18H to 1FH are set aside for the fourth bank of R0–R7.

Default Register Bank

If RAM locations 00–1F are set aside for the four registers banks, which register bank of R0–R7 do we have access to when the 8051 is powered up? The answer is register bank 0; that is, RAM locations from 0 to 7 are accessed with the names R0 to R7 when programming the 8051. Because it is much easier to refer these RAM locations by names such as R0 to R7, rather than by their memory locations.

How to Switch Register Banks

Register bank 0 is the default when the 8051 is powered up. We can switch to the other banks using PSW register. D4 and D3 bits of the PSW are used to select the desired register bank, since they can be accessed by the bit addressable instructions SETB and CLR. For example, "SETB PSW.3" will set PSW.3 = 1 and select the bank register 1.

RS1	RS2	Bank Selected
0	0	Bank0
0	1	Bank1
1	0	Bank2
1	1	Bank3

Stack and its Operations

Stack in the 8051

The stack is a section of a RAM used by the CPU to store information such as data or memory address on temporary basis. The CPU needs this storage area considering limited number of registers.

How Stacks are Accessed

As the stack is a section of a RAM, there are registers inside the CPU to point to it. The register used to access the stack is known as the stack pointer register. The stack pointer in the 8051 is 8-bits wide, and it can take a value of 00 to FFH. When the 8051 is initialized, the SP register contains the value 07H. This means that the RAM location 08 is the first location used for the stack. The storing operation of a CPU register in the stack is known as a PUSH, and getting the contents from the stack back into a CPU register is called a POP.

Pushing into the Stack

In the 8051, the stack pointer (SP) points to the last used location of the stack. When data is pushed onto the stack, the stack pointer (SP) is incremented by 1. When PUSH is executed, the contents of the register are saved on the stack and SP is incremented by 1. To push the registers onto the stack, we must use their RAM addresses. For example, the instruction "PUSH 1" pushes register R1 onto the stack.

Popping from the Stack

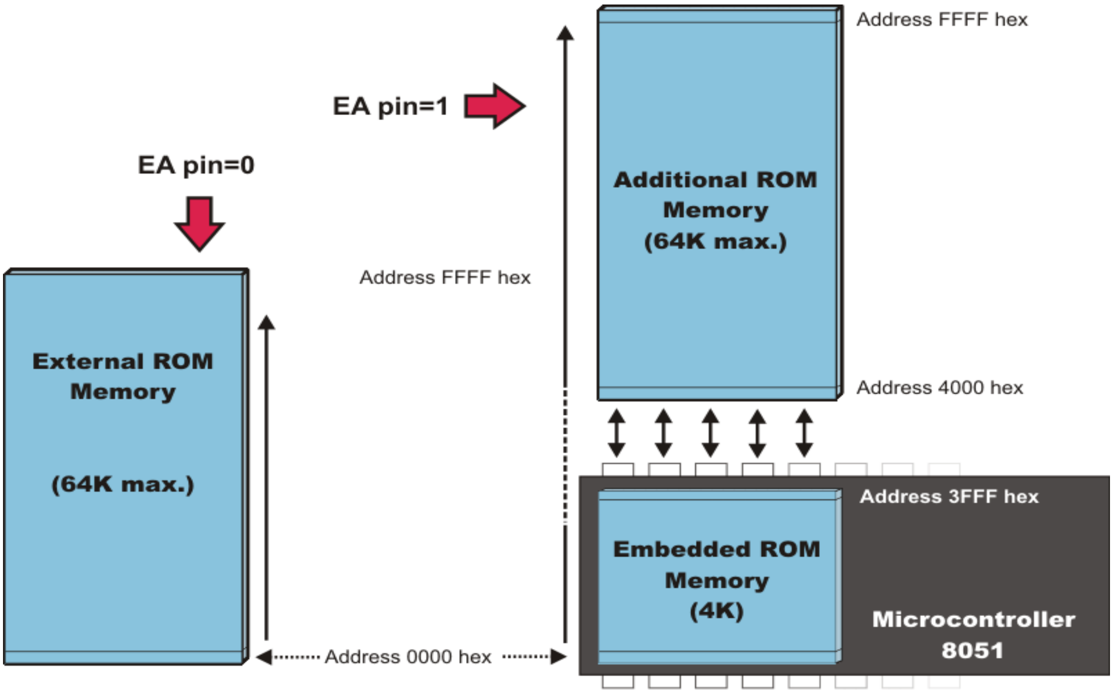
Popping the contents of the stack back into a given register is the opposite to the process of pushing. With every pop operation, the top byte of the stack is copied to the register specified by the instruction and the stack pointer is decremented once.

8051 Memory Organization

The 8051 microcontroller's memory is divided into Program Memory and Data Memory. Program Memory (ROM) is used for permanent saving program being executed, while Data Memory (RAM) is used for temporarily storing and keeping intermediate results and variables.

Program Memory (ROM)
Program Memory (ROM) is used for permanent saving program (CODE) being executed. The memory is read only. Depending on the settings made in compiler, program memory may also used to store a constant variables. The 8051 executes programs stored in program memory only. code memory type specifier is used to refer to program memory.

8051 memory organization allows external program memory to be added.
How does the microcontroller handle external memory depends on the pin EA logical state.



I/O Port usage 8051
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
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

Byte Address	Bit Address								
FF									
F0	F7	F6	F5	F4	F3	F2	F1	F0	B
E0	E7	E6	E5	E4	E3	E2	E1	E0	ACC
D0	D7	D6	D5	D4	D3	D2	-	D0	PSW
B8	-	-	-	BC	BB	BA	B9	B8	IP
B0	B7	B6	B5	B4	B3	B2	B1	B0	P3
A2	AF	-	-	AC	AB	AA	A9	A8	IE
A0	A7	A6	A5	A4	A3	A2	A1	A0	P2
99	Not bit Addressable								SBUF
98	9F	9E	9D	9C	9B	9A	99	98	SCON
90	97	96	95	94	93	92	91	90	P1
8D	Not bit Addressable								TH1
8C	Not bit Addressable								TH0
8B	Not bit Addressable								TL1
8A	Not bit Addressable								TL0
89	Not bit Addressable								TMOD

88	8F	8E	8D	8C	8B	8A	89	88	TCON
87	Not bit Addressable								PCON
83	Not bit Addressable								DPH
82	Not bit Addressable								DPL
81	Not bit Addressable								SP
80	87	87	85	84	83	82	81	80	P0

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.

8051 Microcontroller Special Function Registers (SFRs)

A Special Function Register (or Special Purpose Register, or simply Special Register) is a register within a microprocessor that controls or monitors the various functions of a microprocessor. As the special registers are closely tied to some special function or status of the processor, they might not be directly writable by normal instructions (like add, move, etc.). Instead, some special registers in some processor architectures require special instructions to modify them.

In the 8051, register A, B, DPTR, and PSW are a part of the group of registers commonly referred to as SFR (special function registers). An SFR can be accessed by its name or by its address.

Consider the following two points about the SFR addresses.

A special function register can have an address between 80H to FFH. These addresses are above 80H, as the addresses from 00 to 7FH are the addresses of RAM memory inside the 8051.

Not all the address space of 80 to FF are used by the SFR. Unused locations, 80H to FFH, are reserved and must not be used by the 8051 programmer.

CY	PSW.7	Carry Flag
AC	PSW.6	Auxiliary Carry Flag
F0	PSW.5	Flag 0 available to 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 FLAG
P	PSW.0	Parity FLAG. Set/ cleared by hardware during instruction cycle to indicate even/odd number of 1 bit in accumulator.

In the following example, the SFR registers' names are replaced with their addresses.

CY AC F0 RS1 RS0 OV - P

We can select the corresponding Register Bank bit using RS0 and RS1 bits.

RS1	RS2	Register Bank	Address
0	0	0	00H-07H
0	1	1	08H-0FH
1	0	2	10H-17H
1	1	3	18H-1FH

The Program Status Word (PSW) contains status bits to reflect the current state of the CPU. The 8051 variants provide one special function register, PSW, with this status information. The 8251 provides two additional status flags, Z and N, which are available in a second special function register called PSW1.

Special function registers

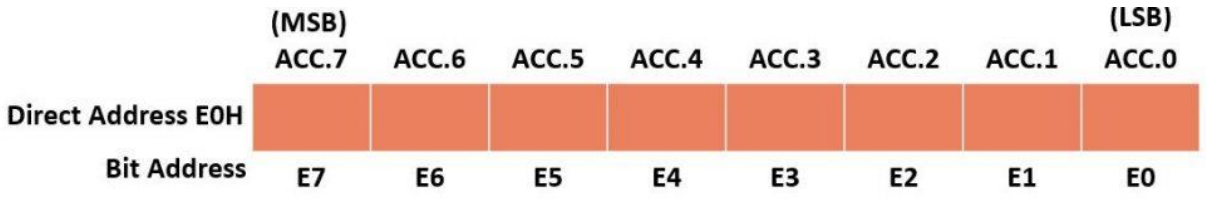
In 8051 micro controller there are 21 Special function registers (SFR) and this includes Register A, Register B, Processor Status Word (PSW), PCON etc etc. There are 21 unique locations for these 21 special function registers and each of these register is of 1 byte size.

The 21 SFR of 8051 Microcontroller are categorized into seven groups these are:

Math or CPU Registers: A and B Register
Status Register: PSW (Program Status Word) Register
Pointer Registers: DPTR (Data Pointer – DPL, DPH) and SP (Stack Pointer) Registers
I/O Port Latches: P0 (Port 0), P1 (Port 1), P2 (Port 2) and P3 (Port 3) Registers
Peripheral Control Registers: PCON, SCON, TCON, TMOD, IE and IP Registers
Peripheral Data Registers: TL0, TH0, TL1, TH1 and SBUF Registers.

Accumulator (A register):

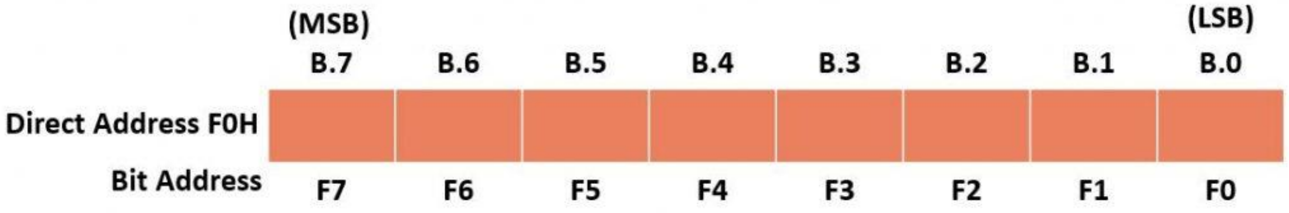
- It is an 8-bit register.
- It holds a data and receives the result of the arithmetic instructions.
- ACC is usually accessed by direct addressing and its physical address is **E0H.** Accumulator is both byte and bit addressable*.* if you want to access the second bit (i.e bit 1), you may use E1H and for third bit E2H and so on.



	(MSB) ACC.7	ACC.6	ACC.5	ACC.4	ACC.3	ACC.2	ACC.1	(LSB) ACC.0
Direct Address E0H								
Bit Address	E7	E6	E5	E4	E3	E2	E1	E0

B Register:

- It is 8-bit register.
- It is bit and byte-addressable register.
- You can access 1-bit or all 8-bits by a physical address F0h. Suppose to access a bit 1, we have to use f1.
- The B register is only used for multiplication and division arithmetic operations.



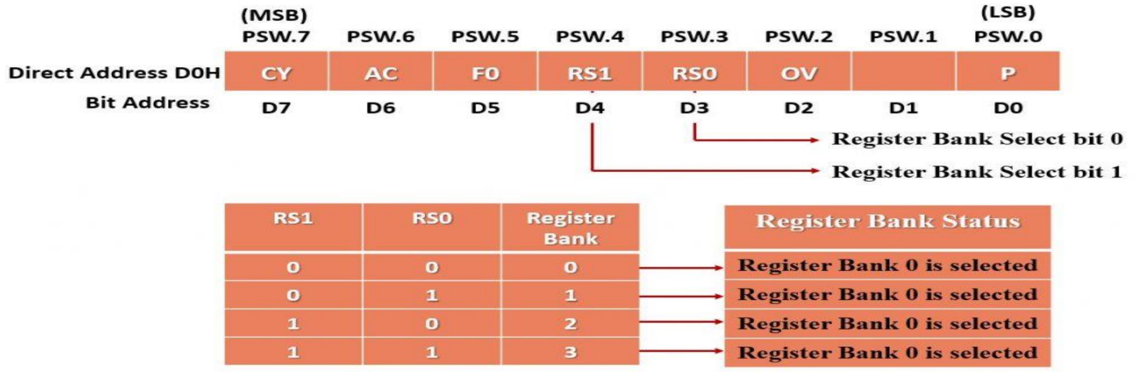
	(MSB) B.7	B.6	B.5	B.4	B.3	B.2	B.1	(LSB) B.0
Direct Address F0H								
Bit Address	F7	F6	F5	F4	F3	F2	F1	F0

PSW (Program Status Word) Register:

The PSW (Program Status Word) Register is also called as Flag Register. It is one of the important SFRs in 8051 microcontrollers. It is also an 8-bit register. It consists of Flag Bits or status bits that reflect the current state of the CPU. This will help the programmer in checking the condition of the result and also make decisions.

PSW flag register is both bit and byte addressable. The physical address of PSW starts from D0H. The individual bits are accessed by using bit address D1, D2 … D7. The two unused bits are user-defined flags. Four of the flags are called conditional flags, which means that they indicate a condition which results after an instruction is executed. These four are CY (Carry), AC (auxiliary carry), P (parity), and OV (overflow).

PSW Register



	(MSB) PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	(LSB) PSW.0
Direct Address D0H	CY	AC	F0	RS1	RS0	OV		P
Bit Address	D7	D6	D5	D4	D3	D2	D1	D0

RS1	RS0	Register Bank	Register Bank Status
0	0	0	Register Bank 0 is selected
0	1	1	Register Bank 0 is selected
1	0	2	Register Bank 0 is selected
1	1	3	Register Bank 0 is selected

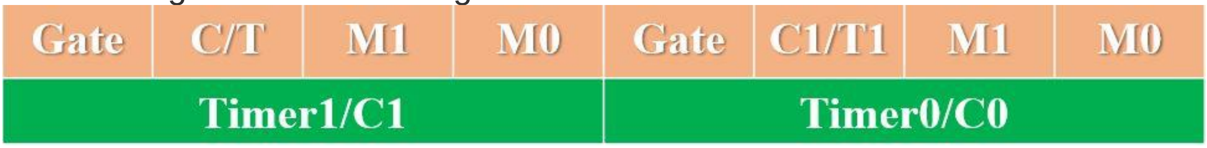
PSW Register Bits

Timer/Counter Registers:

- The 8051 has two 16-bit Programmable timers / counters (Timer 0 – Timer 1).
- Which can be used either as timer to generate a time delay or as counter to count events happening outside the microcontroller.
- The Counters and Timers in 8051 microcontrollers contain two special function registers: TMOD (Timer Mode Register) and TCON (Timer Control Register).

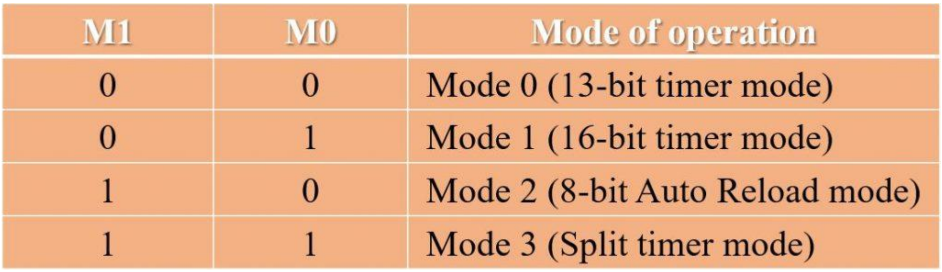
TMODE Registers:

- TMODE register is an 8-bit register.



Gate	C/T	M1	M0	Gate	C1/T1	M1	M0
Timer1/C1				Timer0/C0			

- **Gate:** when Gate control is set. Timer/counter is enable only while the INTx pin is high and the TRx control pin is set. When it is cleared, the timer is enabled whenever the TRx control bit is set.
- **C/T:** The Timer or counter selection. When cleared for timer operation (input from internal system clock). Set for counter operation (input from Tx input pin).
- **Mode selects bits of TMODE register:** The M1 and M0 are mode select bits, which are used to select the timer operations. There are four modes to operate the timers.



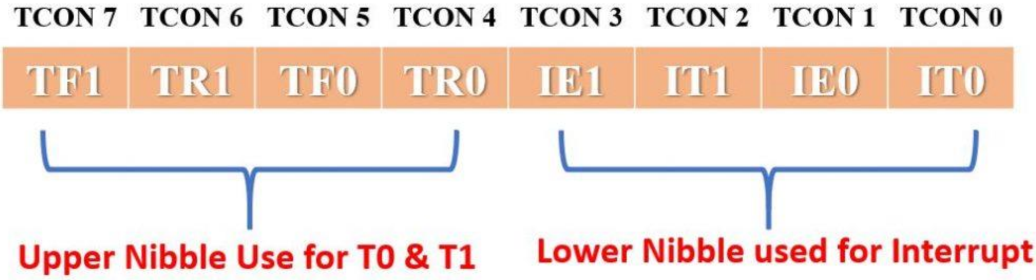
M1	M0	Mode of operation
0	0	Mode 0 (13-bit timer mode)
0	1	Mode 1 (16-bit timer mode)
1	0	Mode 2 (8-bit Auto Reload mode)
1	1	Mode 3 (Split timer mode)

Mode Selection Bits Of TMODE Register

TCON Registers:

- It is Timer control Register.
- It is an 8-bit register.

TCON Register



TCON 7	TCON 6	TCON 5	TCON 4	TCON 3	TCON 2	TCON 1	TCON 0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

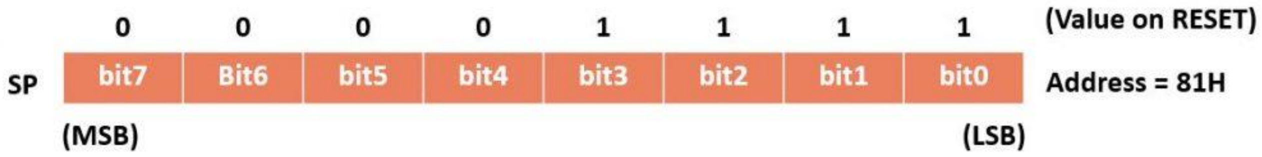
SP (Stack Pointer)

- The stack is a portion of a RAM Used by the CPU to store data or memory address on temporary basis.
- The stack pointer register is used to access the stack is known as SP register. The stack pointer register is 8-bits wide. It can take a value of 00 to FFH. The

RAM memory location 08H is the first location used for the stack. When the 8051 microcontroller is initialized, the SP register contains the value 07H.

- If we want to store CPU register data into the stack this is known as PUSH operation and if we getting the data from stack back into a CPU register this is known as a POP operation.

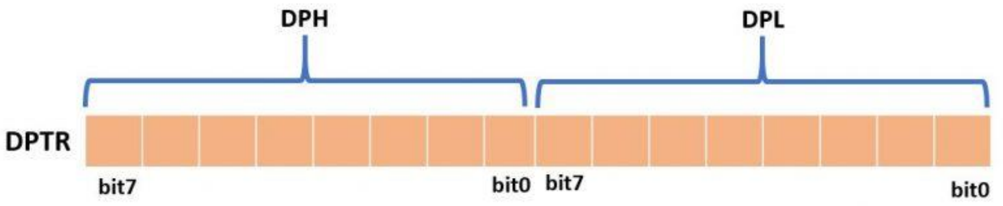
SP Register

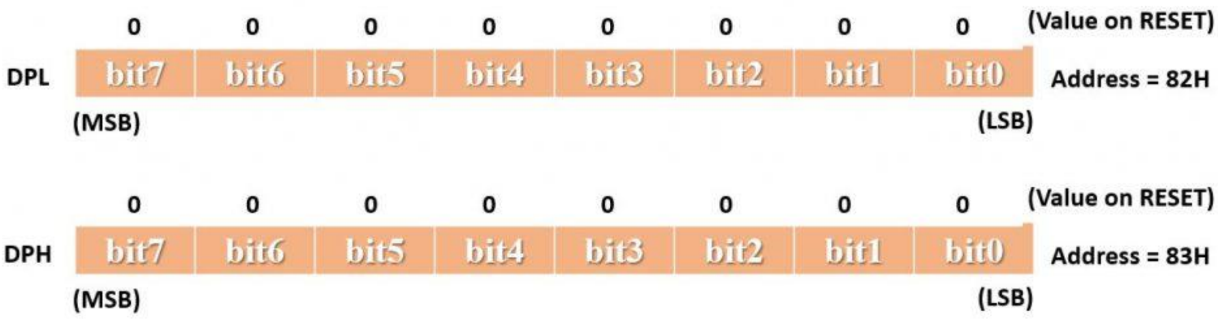


	0	0	0	0	1	1	1	1	(Value on RESET)
SP	bit7	Bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = 81H
	(MSB)							(LSB)	

DP (Data pointer register):

- The data pointer is a 16-bit register used to hold the 16-bit address of data memory. This can also be used as two 8-bit register namely DPH and DPL. Data Pointer can be used as a single 16-bit register (as DPTR) or two 8-bit registers (as DPL and DPH).
- The 8-bit data pointers are used for accessing internal RAM and SFR. The 16-bit data pointer is used for accessing external data memory.
- The contents of data pointer are programmable using instructions. It is used by the 8051 to access external memory using the address indicated by DPTR.





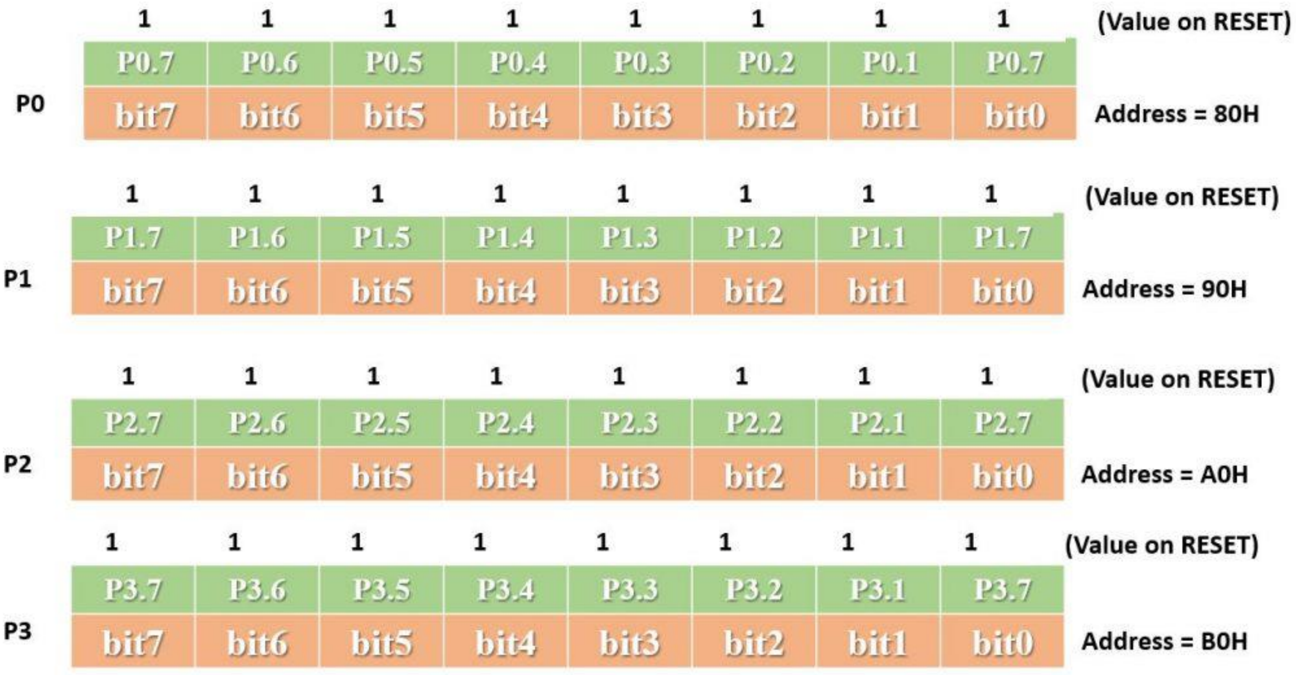
	0	0	0	0	0	0	0	0	(Value on RESET)
DPL	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = 82H
	(MSB)							(LSB)	

	0	0	0	0	0	0	0	0	(Value on RESET)
DPH	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = 83H
	(MSB)							(LSB)	

Data Pointer Register

Port Registers:

8051 microcontrollers have 4 bidirectional I/O ports. Port 0, Port 1, Port 2 and Port 3 (P0, P1, P2 and P3). Which can be work as input or output port. Each port having 8-bits. Hence, total 32 input/output pins allow the microcontroller to communicate with outside world means this port allow to be connected with the peripheral devices.



	1	1	1	1	1	1	1	1	(Value on RESET)
	P0.7	P0.6	P0.5	P0.4	P0.3	P0.2	P0.1	P0.7	
P0	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = 80H
	1	1	1	1	1	1	1	1	(Value on RESET)
	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.7	
P1	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = 90H
	1	1	1	1	1	1	1	1	(Value on RESET)
	P2.7	P2.6	P2.5	P2.4	P2.3	P2.2	P2.1	P2.7	
P2	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = A0H
	1	1	1	1	1	1	1	1	(Value on RESET)
	P3.7	P3.6	P3.5	P3.4	P3.3	P3.2	P3.1	P3.7	
P3	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	Address = B0H

Input output Port Register

PC (program Counter):

It is a 16-bit register. It is use to hold the address of the memory location from where the next instruction to be fetched.

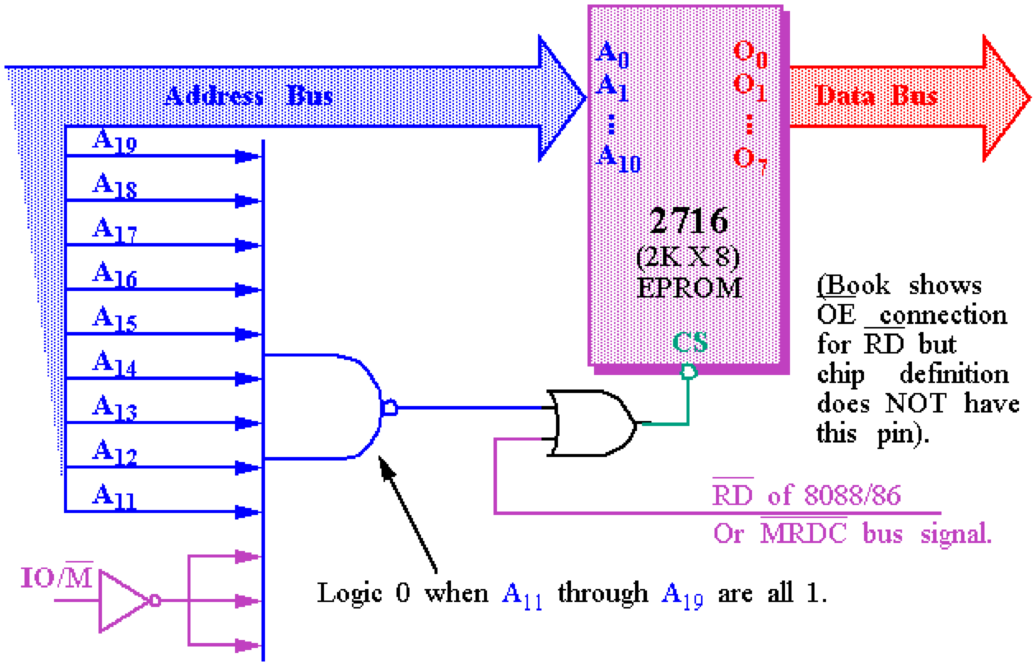
when the 8051 initializes PC starts at 0000h and it is automatically is incremented every time after an instruction is executed. (So in this way PC maintain the sequence of program execution).

Due to this the width of the PC decides the max program length in bytes.

Memory Address Decoding

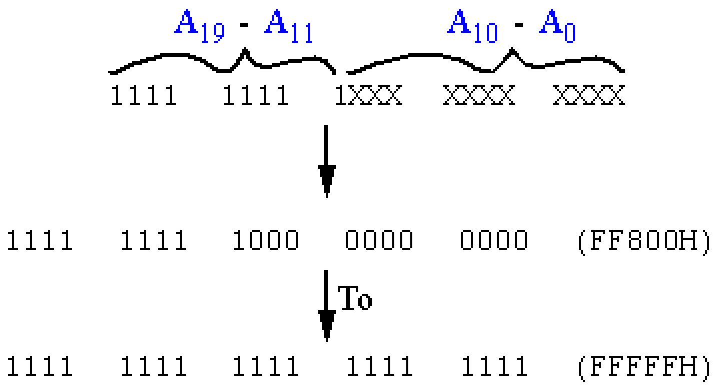
- The processor can usually address a memory space that is much larger than the memory space covered by an individual memory chip.
- In order to splice a memory device into the address space of the processor, decoding is necessary.
- For example, the 8088 issues 20-bit addresses for a total of **1MB** of memory address space.
- However, the BIOS on a 2716 EPROM has only 2KB of memory and 11 address pins.
- A decoder can be used to decode the additional 9 address pins and allow the EPROM to be placed in **any** 2KB section of the 1MB address space.

Memory Address Decoding


Address Bus
A_{19}
A_{18}
A_{17}
A_{16}
A_{15}
A_{14}
A_{13}
A_{12}
A_{11}
A_0
A_1
A_{10}
O_0
O_1
O_7
2716
(2K X 8)
EPROM
CS
Data Bus
(Book shows OE connection for RD but chip definition does NOT have this pin).
$\overline{RD}$ of 8088/86
Or $\overline{MRDC}$ bus signal.
IO/$\overline{M}$
Logic 0 when A_{11} through A_{19} are all 1.

Memory Address Decoding

- *To determine the address range that a device is mapped into:*

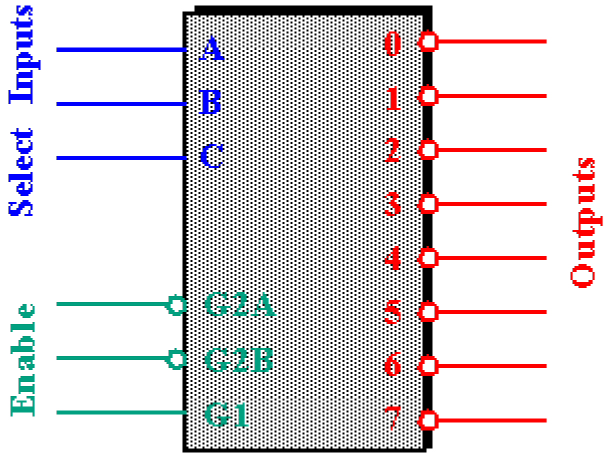

A19 - A11
A10 - A0
1111 1111 1XXX XXXX XXXX
1111 1111 1000 0000 0000 (FF800H)
To
1111 1111 1111 1111 1111 (FFFFFH)

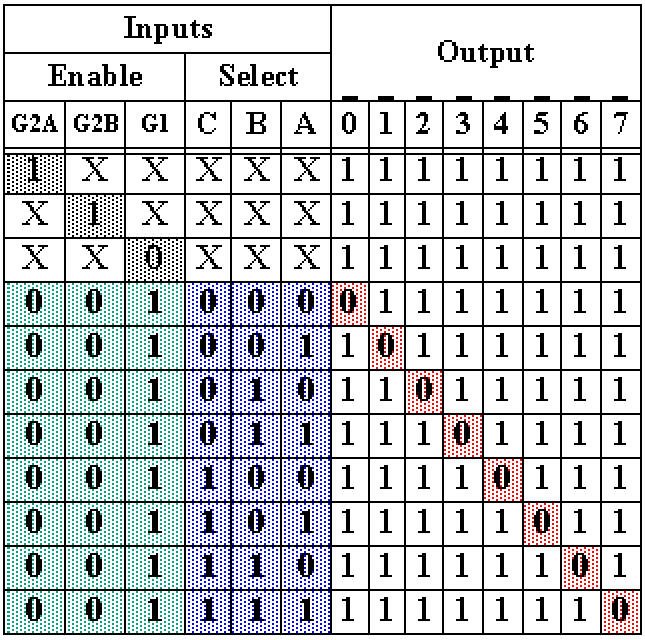
- *This 2KB memory segment maps into the* **reset location** *of the 8086/8088 (FFFF0H).*

- *NAND gate decoders are not often used.*
    - *Rather the 3-to-8 Line Decoder (74LS138) is more common.*

Memory Address Decoding

- *The 3-to-8 Line Decoder (74LS138)*

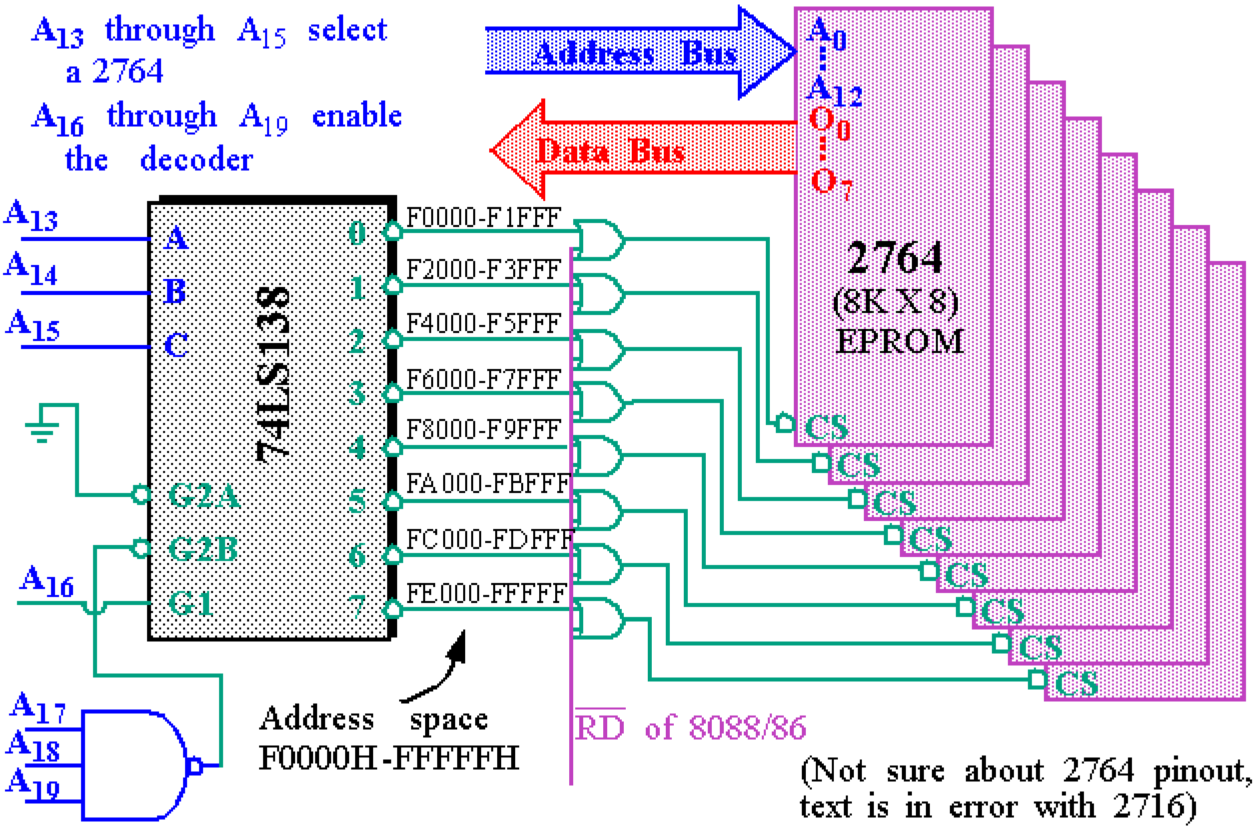




Inputs						Output							
Enable			Select										
G2A	G2B	G1	C	B	A	$\overline{0}$	$\overline{1}$	$\overline{2}$	$\overline{3}$	$\overline{4}$	$\overline{5}$	$\overline{6}$	$\overline{7}$
1	X	X	X	X	X	1	1	1	1	1	1	1	1
X	1	X	X	X	X	1	1	1	1	1	1	1	1
X	X	0	X	X	X	1	1	1	1	1	1	1	1
0	0	1	0	0	0	0	1	1	1	1	1	1	1
0	0	1	0	0	1	1	0	1	1	1	1	1	1
0	0	1	0	1	0	1	1	0	1	1	1	1	1
0	0	1	0	1	1	1	1	1	0	1	1	1	1
0	0	1	1	0	0	1	1	1	1	0	1	1	1
0	0	1	1	0	1	1	1	1	1	1	0	1	1
0	0	1	1	1	0	1	1	1	1	1	1	0	1
0	0	1	1	1	1	1	1	1	1	1	1	1	0

- *Note that all three Enables (G2A, G2B, and G1) must be active, e.g. low, low and high, respectively.*
- *Each output of the decoder can be attached to an 2764 EPROM (8K X 8).*

Memory Address Decoding



- *The EPROMs cover a 64KB section of memory.*

Memory Address Decoding

- *Yet a third possibility is a* **PLD** *(Programmable Logic Device).*
- *PLDs come in three varieties:*
- *PLA (Programmable Logic Array)*
- *PAL (Programmable Array Logic)*
- *GAL (Gated Array Logic)*

- *PLDs have been around since the mid-1970s but have only recently appeared in memory systems (PALs have replaced PROM address decoders).*

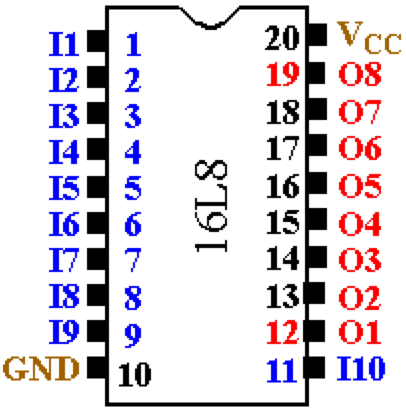
- *PALs and PLAs are* **fuse-programmed** *(like the PROM).*
  - *Some are erasable (like the EPROM).*

- *A PAL example (16L8) is shown in the text and is commonly used to decode the memory address, particularly for 32-bit addresses generated by the 80386DX and above.*

Memory Address Decoding

- *AMD 16L8 PAL decoder.*
  - *It has 10 fixed inputs (Pins 1-9, 11), two fixed outputs (Pins 12 and 19) and 6 pins that can be either (Pins 13-18).*

Programmed to decode address lines A_{19} - A_{13} onto 8 outputs.



```
;pins  1    2    3    4    5    6    7    8   9   10
      A19  A18  A17  A16  A15  A14  A13  NC  NC  GND
;pins  11   12   13   14   15   16   17   18  19  20
       NC   O8   O7   O6   O5   O4   O3   O2  O1  VCC

Equations:
 /O1 = A19 * A18 * A17 * A16 * /A15 * /A14 * /A13
 /O2 = A19 * A18 * A17 * A16 * /A15 * /A14 *  A13
 /O3 = A19 * A18 * A17 * A16 * /A15 *  A14 * /A13
 /O4 = A19 * A18 * A17 * A16 * /A15 *  A14 *  A13
 /O5 = A19 * A18 * A17 * A16 *  A15 * /A14 * /A13
 /O6 = A19 * A18 * A17 * A16 *  A15 * /A14 *  A13
 /O7 = A19 * A18 * A17 * A16 *  A15 *  A14 * /A13
 /O8 = A19 * A18 * A17 * A16 *  A15 *  A14 *  A13
```

- AND/NOR device with logic expressions (outputs) with up to 16 ANDed inputs and 7 ORed product terms.

Interfacing External Memory with 8051 Microcontroller

It is always good to have an option to expand the capabilities of a Microcontroller, whether it is in terms of Memory or IO or anything else. Such expansion will be useful to avoid design throttling. We have seen that a typical 8051 Microcontroller has 4KB of ROM and 128B of RAM (most modern 8051 Microcontroller variants have 8K ROM and 256B of RAM).

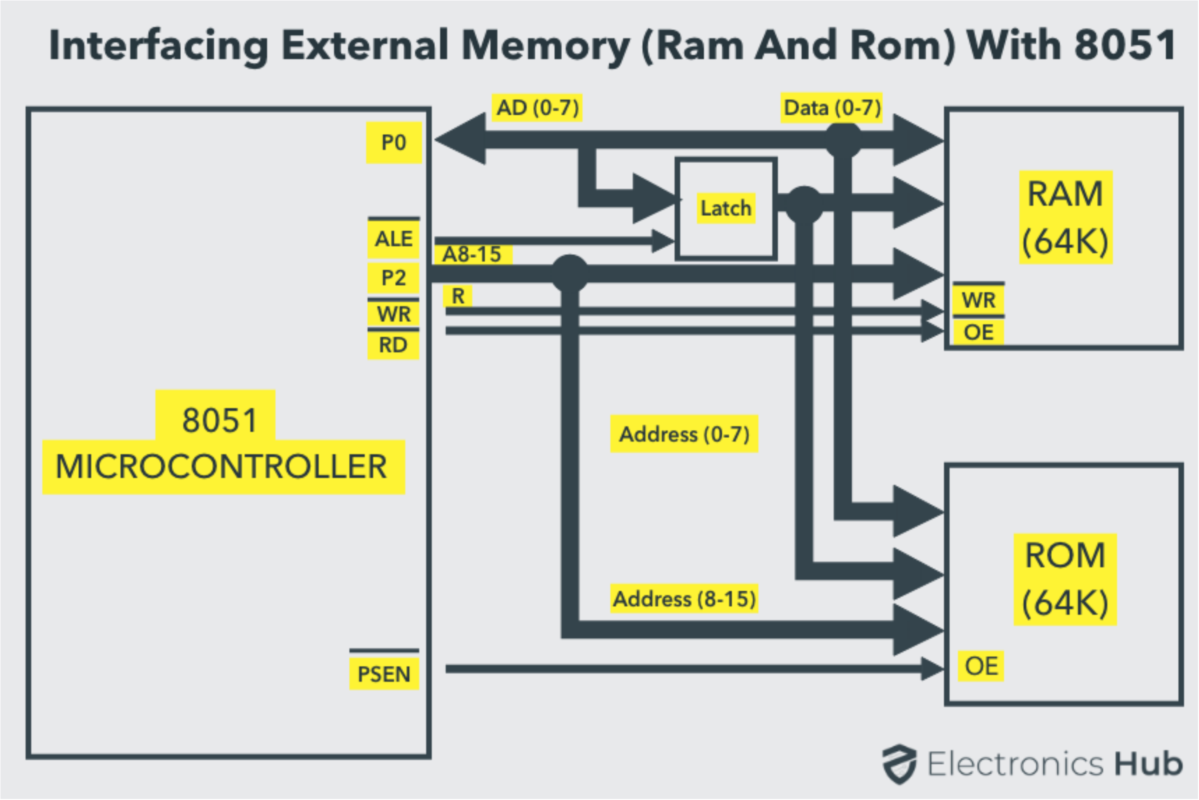
The designer of an 8051 Microcontroller based system is not limited to the internal RAM and ROM present in the 8051 Microcontroller. There is a provision of connecting both external RAM and ROM i.e., Data Memory and Program.

The reason for interfacing external Program Memory or ROM is that complex programs written in high – level languages often tend to be larger and occupy more memory.

Another important reason is that chips like 8031 or 8032, which doesn't have any internal ROM, have to be interfaced with external ROM.

A maximum of 64KB of Program Memory (ROM) and Data Memory (RAM) each can be interface with the 8051 Microcontroller.

The following image shows the block diagram of interfacing 64KB of External RAM and 64KB of External ROM with the 8051 Microcontroller.



8051 Interfacing External RAM and ROM

An important point to remember when interfacing external memory with 8051 Microcontroller is that Port 0 (P0) cannot be used as an IO Port as it will be used for multiplexed address and data bus (A0 – A7 and D0 – D7). Not always, but Port 2 may be used as higher byte of the address bus.

8051 Addressing Modes

In this tutorial, we have seen the 8051 Microcontroller Memory Organization, Program Memory, Data Memory, Internal ROM and RAM and how to interface external Memory (ROM and RAM) with 8051 Microcontroller.

In this section, we will see different addressing modes of the 8051 microcontrollers. 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 µsto 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 Register Addressing 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 these two instructions 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.

```
MOV 80H, R6;
MOV R2, 45H;
MOV R0, 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.

```
MOV 0E5H, @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.

```
MOV XA, @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.
MOV CA, @A+PC;
MOV CA, @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.