1. Write any two differences between microprocessor and microcontroller.
2. What is a microcontroller and what are the characteristics of microcontroller
3. Give the overview of 8051 microcontroller family
4. Explain the architecture of 8051 with a neat diagram.

Microprocessor

As its name implies, it is a processing device that converts data into information based on some sets of instructions. It is a very compact electronic chip due to which it is referred to as the **microprocessor**.

In other words, a processing device implemented on a single chip is called a microprocessor. A microprocessor is the most crucial component of a computer or any other computing device. Because, it is entirely responsible for processing data based on instructions to produce information.

In microcomputers, the microprocessor is used as the CPU (Central Processing Unit). A typical microprocessor consists of two major parts namely ALU (Arithmetic Logic Unit) and CU (Control Unit). Intel 8085 or 8086 processing chips are the examples of microprocessors.

Modern microprocessors consist of a small memory unit (cache memory) in addition to the ALU and CU. Now−a−days, microprocessors are being widely used in several applications such as desktop publishing, power plant control, millimeters, medical instruments, etc.

Microcontroller

A **microcontroller** is an electronic system which consists of a processing element, a small memory (RAM, ROM, EPROM), I/O ports, etc. on a single chip. Therefore, a microcontroller is a tiny resemblance of a microcomputer. It is a quite small and low−cost electronic device which is used in several electronic appliances as the main functioning device.

In electronic systems such washing machines, air conditioners, refrigerators, etc., microcontrollers are used to automate the operation of the device based on user's instructions.

Hence, a microcontroller is the backbone of all embedded systems like microwave oven, washing machine, smart refrigerators, etc.

MICROPROCESSOR	MICROCONTROLLER
Center of a computer system.	Center of embedded system.
Memory and I/O components are external to it.	Memory and I/O components are internal to it.
Large Circuit	Smaller Circuit
Not compatible with compact systems	Compatible with compact systems.
Higher cost	Lower Cost
High Power Consumption	Low Power Consumption
Mostly don't have power features	Mostly have power features.
Mainly present in personal computers.	Mainly present in washing machines, music players, and embedded systems.
Less number of registers.	More number of registers.
Follows Von Neumann model	Follows Harvard architecture
Made on a silicon-based integrated chip.	Byproduct microprocessors and peripherals.
RAM, ROM, and other peripherals are absent.	RAM, ROM, and other peripherals are present.
Has an external bus to interface with devices.	Uses an internal controlling bus for communication.
Has a high speed.	Speed depends on the architecture.
Ideal for general purpose to handle more data.	Ideal for the specific applications.
Complex and Expensive	Simple and affordable
Requires more instructions	Requires less instructions

Characteristics of a microcontroller:

1. It is a small computer. It has processor and some other components.
2. Used in automatically controlled devices.
3. Used in Embedded systems.
4. It has less computational capacity than microprocessor. So it is used for simpler tasks only.
5. Do not have math coprocessors.
6. Perform tasks – fetch, decode and execute.
7. It has memory, both RAM and ROM with some other I/O devices too.
8. Power consumption is less in microcontroller.
9. Optimize interrupt latency.
10. Bit manipulation is powerful.

11. Used to handle real time tasks and they are single programmed, self-sufficient and task oriented.

Types of Microcontrollers

Microcontrollers are divided into various categories based on memory, architecture, bits and instruction sets. Following is the list of their types −

Bit

Based on bit configuration, the microcontroller is further divided into three categories.

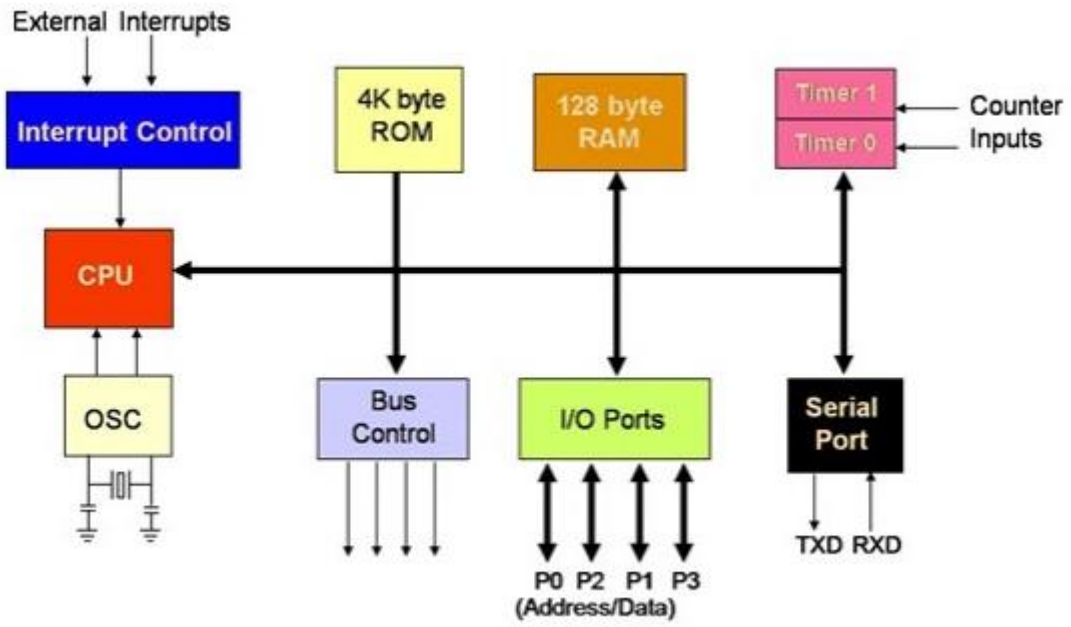
1. 8-bit microcontroller − This type of microcontroller is used to execute arithmetic and logical operations like addition, subtraction, multiplication division, etc. For example, Intel 8031 and 8051 are 8 bits microcontroller.
2. 16-bit microcontroller − This type of microcontroller is used to perform arithmetic and logical operations where higher accuracy and performance is required. For example, Intel 8096 is a 16-bit microcontroller.
3. 32-bit microcontroller − This type of microcontroller is generally used in automatically controlled appliances like automatic operational machines, medical appliances, etc.

Applications of Microcontrollers

1. Microcontrollers are widely used in various different devices such as −
2. Light sensing and controlling devices like LED.
3. Temperature sensing and controlling devices like microwave oven, chimneys.
4. Fire detection and safety devices like Fire alarm.
5. Measuring devices like Volt Meter.

8051 Microcontroller Architecture

Let's see the internal architecture of 8051 Microcontroller represented in form of block diagram as shown below:

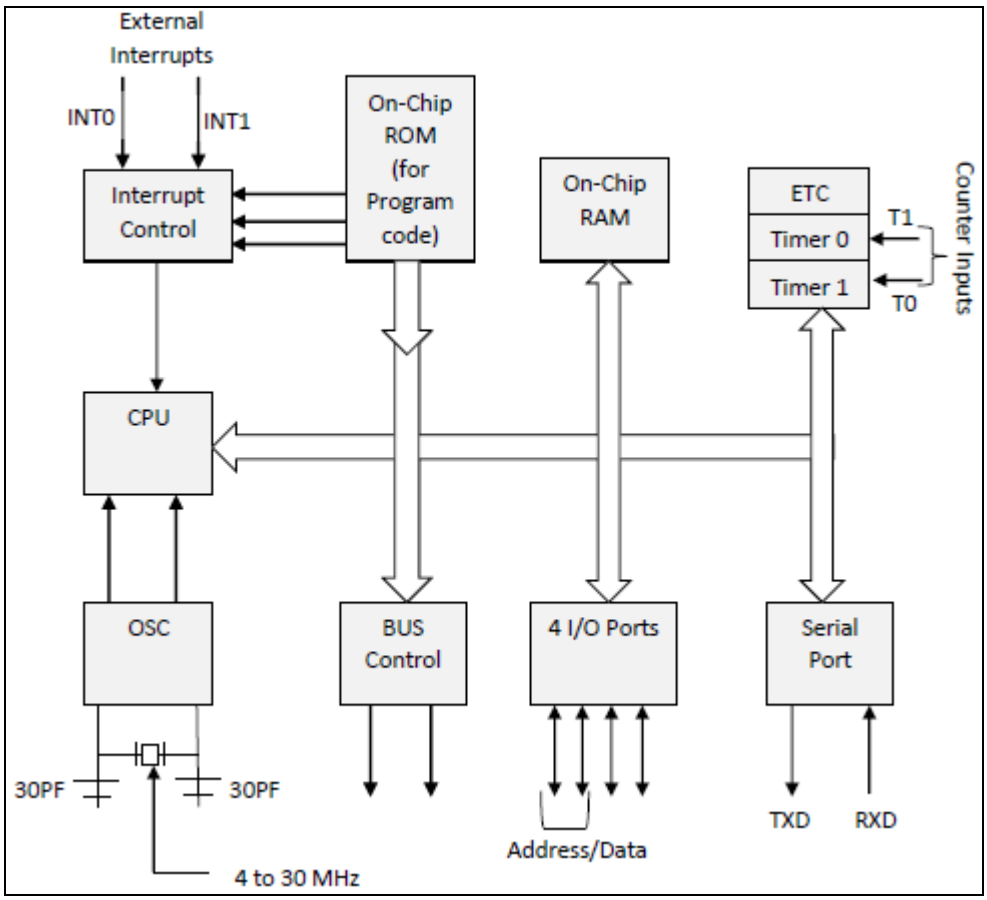


8051 microcontroller is designed by Intel in 1981. It is an 8-bit microcontroller. It is built with 40 pins DIP (dual inline package), 4kb of ROM storage and 128 bytes of RAM storage, 2 16-bit timers. It consists of are four parallel 8-bit ports, which are programmable as well as addressable as per the requirement. An on-chip crystal oscillator is integrated in the microcontroller having crystal frequency of 12 MHz.

Let us now discuss the architecture of 8051 Microcontroller.

In the following diagram, the system bus connects all the support devices to the CPU. The system bus consists of an 8-bit data bus, a 16-bit address bus and bus control signals. All other devices like program memory, ports, data memory,

serial interface, interrupt control, timers, and the CPU are all interfaced together through the system bus.



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 manages the different types of registers available in 8051 microcontroller.

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 interrupt 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.

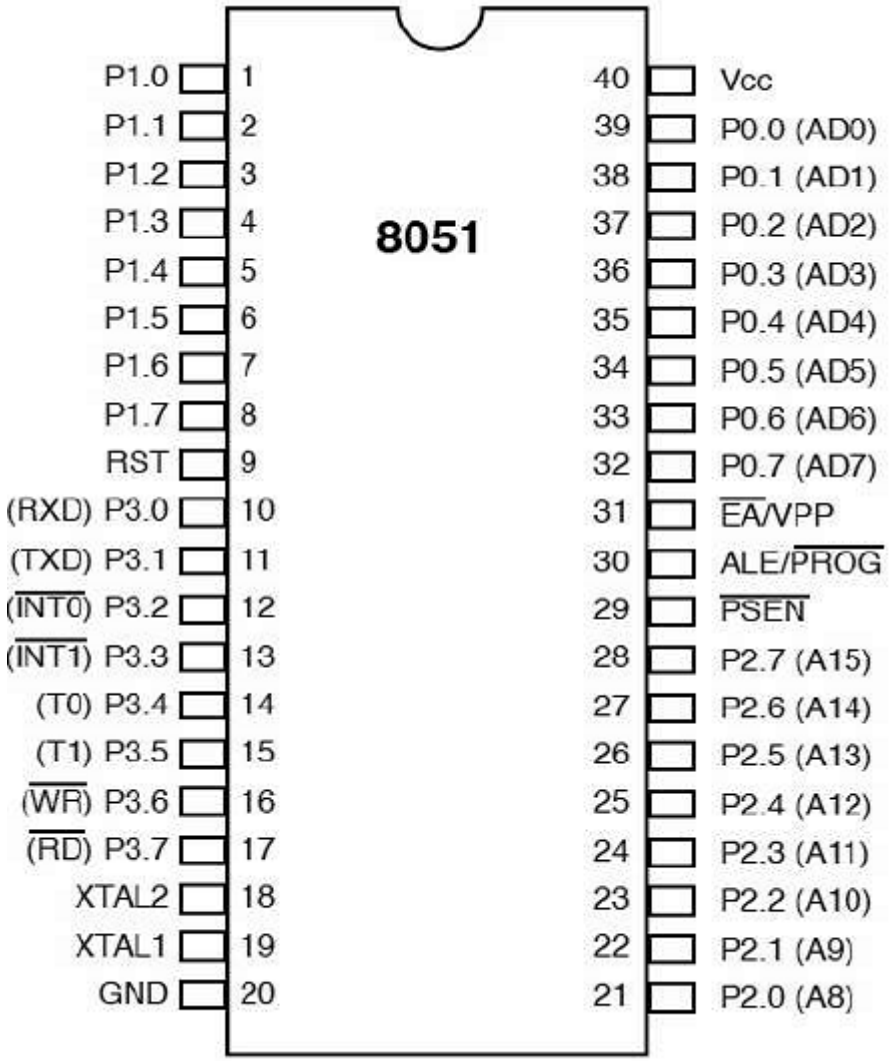
Types of buses in 8051 Microcontroller:

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.

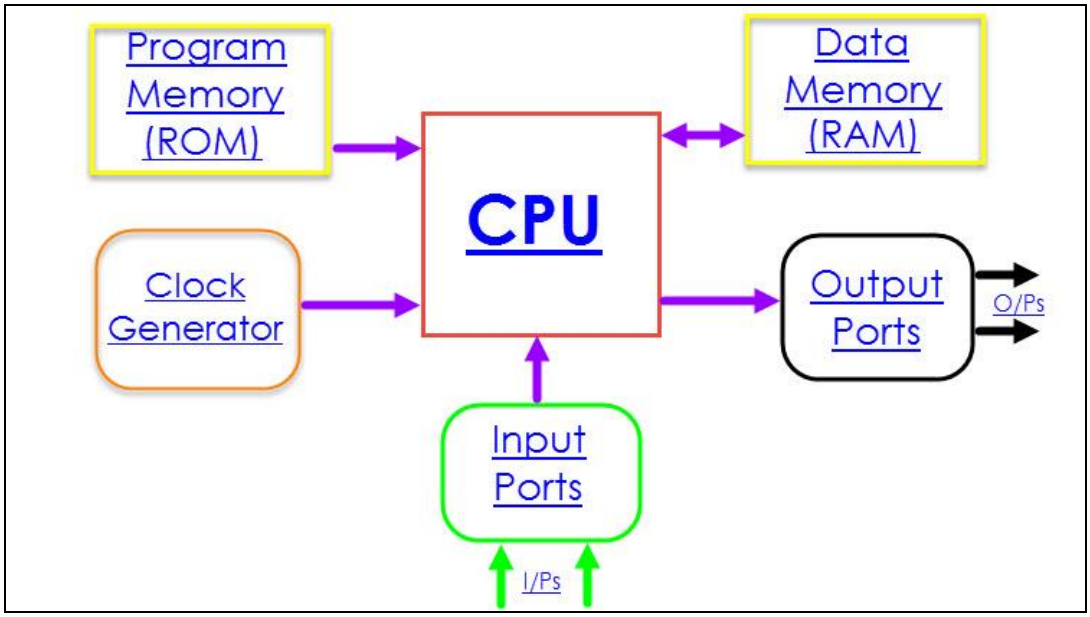
Oscillator: As the microcontroller is digital circuit therefore it needs timer for their operation. To perform timer operation inside microcontroller it required externally connected or on-chip oscillator. Microcontroller is used inside an embedded system for managing the function of devices. Therefore,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

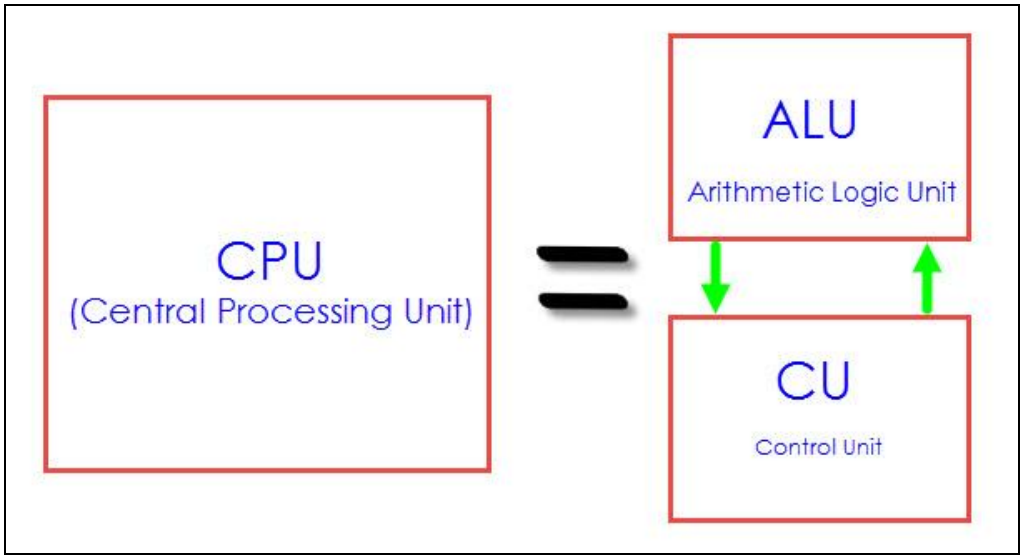
The pin diagram of 8051 microcontroller looks as follows −

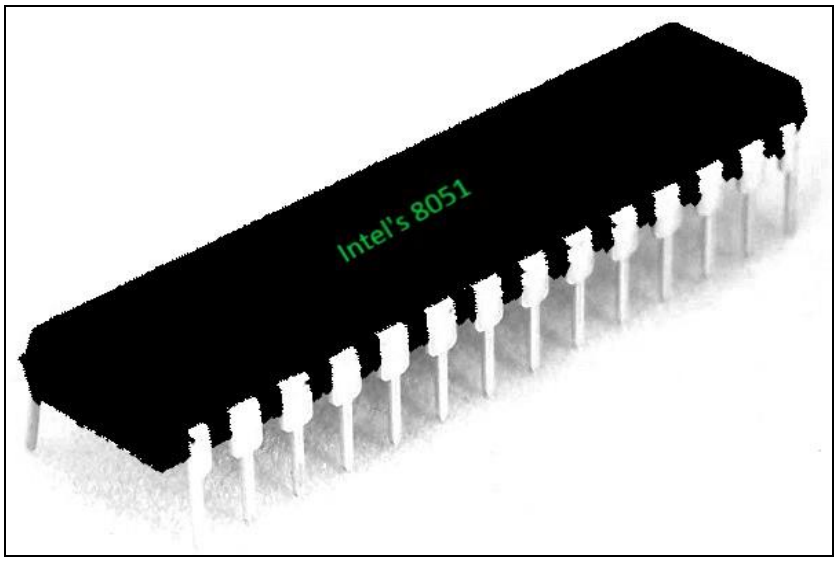


- **Pins 1 to 8** − these pins are known as Port 1. This port doesn't serve any other functions. It is internally pulled up, bi-directional I/O port.
- **Pin 9** − It is a RESET pin, which is used to reset the microcontroller to its initial values.
- **Pins 10 to 17** − these pins are known as Port 3. 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.




CPU
(Central Processing Unit)
=
ALU
Arithmetic Logic Unit
CU
Control Unit


Intel's 8051

THE PROGRAM COUNTER AND ROM SPACE IN THE 8051

In this section we examine the role of the program counter (PC) register in executing an 8051 program. We also discuss ROM memory space for various 8051 family members.

Program counter in the 8051

Another important register in the 8051 is the PC (program counter). The program counter points to the address of the next instruction to be executed. As the CPU fetches the opcode from the program ROM, the program counter is incremented to point to the next instruction. The program counter in the 8051 is 16 bits wide. This means that the 8051 can access program addresses 0000 to FFFFH, a total of 64K bytes of code. However, not all members of the 8051 have the entire 64K bytes of on-chip ROM installed, as we will see soon. Where does the 8051 wake up when it is powered? We will discuss this important topic next.

Where the 8051 wakes up when it is powered up

One question that we must ask about any microcontroller (or microprocessor) is: At what address does the CPU wake up upon applying power to it? Each microprocessor is different. In the case of the 8051 family (that is, all members regardless of the maker and variation), the microcontroller wakes up at memory address 0000 when it is powered up. By powering up we mean applying V_{cc} to the RESET pin as discussed in Chapter 4. In other words, when the 8051 is powered up,. the PC (program counter) has the value of 0000 in it. This means that it expects the first opcode to be stored at ROM address OOOOH. For this reason in the 8051 system, the first opcode must be burned into memory location OOOOH of program ROM since this is where it looks for the first instruction when it is booted. We achieve this by the ORG statement in the source program as shown earlier. Next, we discuss the step-by-step action of the program counter in fetching and executing a sample program.

Placing code in program ROM

To get a better understanding of the role of the program counter in fetching and executing a program, we examine the action of the program counter as each instruction is fetched and executed. First, we examine once more the list file of the sample program and how the code is placed in the ROM of an 8051 chip. As we can see, the opcode and operand for each instruction are listed on the left side of the list file.

```
1 0000                ORG 0H          ;start at location 0
2 0000 7D25           MOV R5,#25H     ;load 25H into R5
3 0002 7F34           MOV R7,#34H     ;load 34H into R7
4 0004 7400           MOV A,#0        ;load 0 into A
5 0006 2D             ADD A,R5        ;add contents of R5 to A
                                      ;now A = A + R5
6 0007 2F             ADD A,R7        ;add contents of R7 to A
                                      ;now A = A + R7
7 0008 2412           ADD A,#12H      ;add to A value 12H
                                      ;now A = A + 12H
8 000A 80FE HERE:     SJMP HERE       ;stay in this loop
9 000C                END             ;end of asm source file
```

Program21: List File ROM Address

	Machine Language	Assembly Language
0000	7D25	MOV R5,#25H
0002	7F34	MOV R7,#34H
0004	7400	MOV A,#0
0006	2D	ADD A,R5
0007	2F	ADD A,R7
0008	2412	ADD A,#12H
000A	80FE	HERE: SJMP HERE

After the program in is burned into ROM of an 8051 family member such as 8751 or AT8951 or DS5000, the opcode and operand are placed in ROM memory locations starting at 0000 as shown in the list below.

Program 2-1: ROM Contents

Address	Code
0000	7D
0001	25
0002	7F
0003	34
0004	74
0005	00
0006	2D
0007	2F
0008	24
0009	12
000A	80
000B	FE

The list shows that address 0000 contains 7D, which is the opcode for moving a value into register R5, and address 0001 contains the operand (in this case 25H) to be moved to R5. Therefore, the instruction "MOV R5,#25H" has a machine code of "7D25″, where 7D is the opcode and 25 is the operand. Similarly, the machine code "7F34″ is located in memory locations 0002 and 0003 and represents the opcode and the operand for the instruction "MOV R7,#34H". In the same way, machine code "7400″ is located in memory locations 0004 and 0005 and represents the opcode and the operand for the instruction "MOV A, #0″. The memory location 0006 has the opcode of 2D, which is the opcode for the instruction "ADD A, R5″ and memory location 0007 has the content 2F, which is the opcode for the "ADD A, R7″ instruction. The opcode for the instruction "ADD A, #12H" is located at address 0008 and the operand 12H at address 0009. The memory location OOOA has the opcode for the SJMP instruction and its target address is located in location OOOB. The reason the target address is FE is explained in the next chapter.

Executing a program byte by byte

Assuming that the above program is burned into the ROM of an 8051 chip (or 8751, AT8951, or DS5000), the following is a step-by-step description of the action of the 8051 upon applying power to it.

1. When the 8051 is powered up, the PC (program counter) has 0000 and starts to fetch the first opcode from location 0000 of the program ROM. In the case of the above program the first opcode is 7D, which is the code for moving an operand to R5. Upon executing the opcode, the CPU fetches the value 25 and places it in R5. Now one instruction is finished. Then the program counter is incremented to point to 0002 (PC = 0002), which contains opcode 7F, the opcode for the instruction "MOV R7 , . .".
2. Upon executing the opcode 7F, the value 34H is moved into R7. Then the program counter is incremented to 0004.
3. ROM location 0004 has the opcode for the instruction "MOV A, #0″. This instruction is executed and now PC = 0006. Notice that all the above instruc tions are 2-byte instructions; that is, each one takes two memory locations.
4. Now PC = 0006 points to the next instruction, which is "ADD A, R5″. This is a 1-byte instruction. After the execution of this instruction, PC = 0007.

5. The location 0007 has the opcode 2F, which belongs to the instruction "ADD A,R7". This also is a 1-byte instruction. Upon execution of this instruction, PC is incremented to 0008. This process goes on until all the instructions are fetched and executed. The fact that the program counter points at the next instruction to be executed explains why some microprocessors (notably the x86) call the program counter the *instruction pointer*.

ROM memory map in the 8051 family

As we saw in the last chapter, some family members have only 4K bytes of on-chip ROM (e.g., 8751, AT8951) and some, such as the AT89C52, have 8K bytes of ROM. Dallas Semiconductor's DS5000-32 has 32K bytes of on-chip ROM. Dallas Semiconductor also has an 8051 with 64K bytes of on-chip ROM. 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 the 8051 is a 16-bit register (0000 to FFFF address range). It must be noted that while the first location of program ROM inside the 8051 has the address of 0000, the last location can be different depending on the size of the ROM on the chip. Among the 8051 family members, the 8751 and AT8951 have 4K bytes of on-chip ROM. This 4K bytes of ROM memory has memory addresses of 0000 to OFFFH. Therefore, the first location of on-chip ROM of this 8051 has an address of 0000 and the last location has the address of OFFFH. Look at Example 2-1 to see how this is computed.

Example 2-1

Find the ROM memory address of each of the following 8051 chips.

(a) AT89C51 with 4KB (b) DS89C420 with 16KB (c) DS5000-32 with 32KB

Solution:

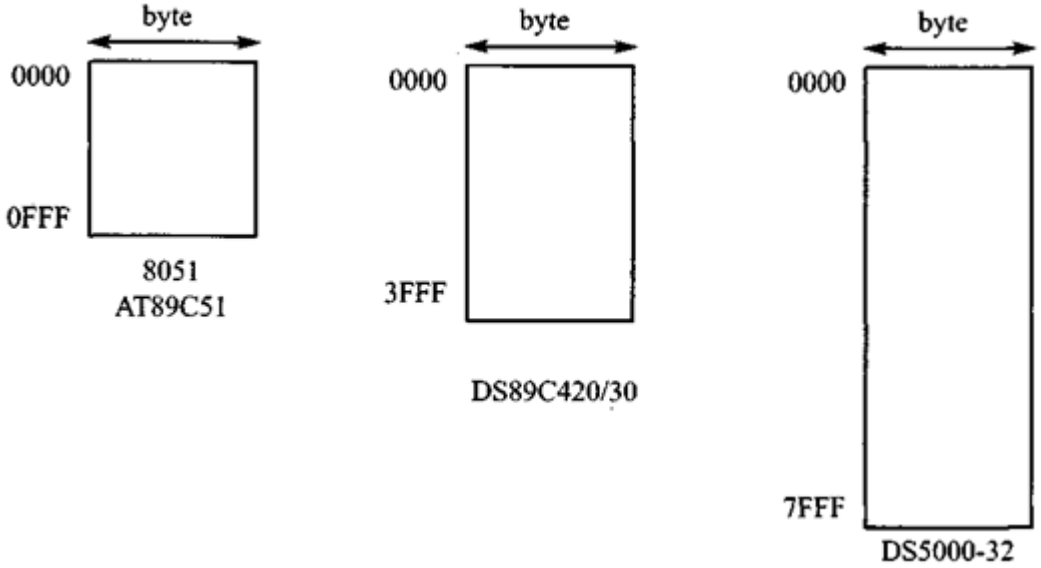


Figure 2-3. 8051 On-Chip ROM Address Range

(a) With 4K bytes of on-chip ROM memory space, we have 4096 bytes (4 x 1024 = 4096). This maps to address locations of 0000 to OFFFH. Notice that 0 is always the first location, (b) With 16K bytes of on-chip ROM memory space, we have 16,384 bytes (16 x 1024 = 16,384), which gives 0000 – 3FFFH. (c) With 32K bytes we have 32,768 bytes (32 x 1024 = 32,768). Converting 32,768 to hex, we get 8000H; therefore, the memory space is 0000 to 7FFFH.

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

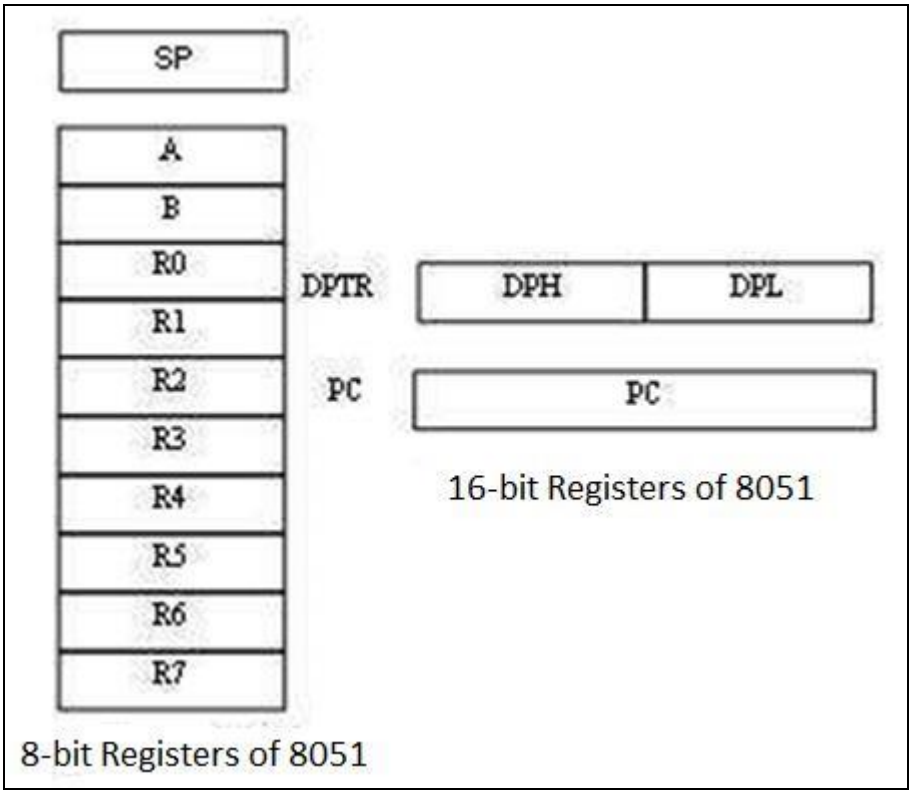
After executing this instruction, the accumulator will contain the value 30. Thus "R" registers are very important auxiliary or **helper registers**. The Accumulator alone would not be very

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.



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.

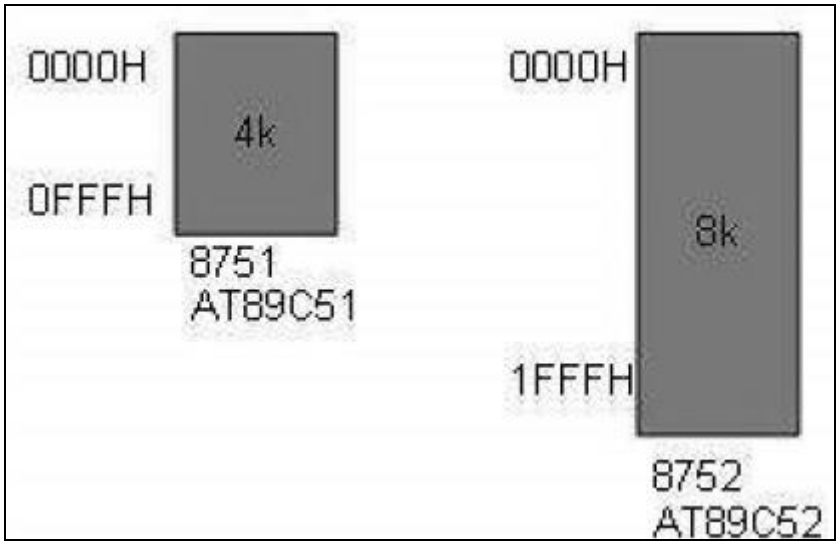
This order of operation is important. SP will be initialized to 07h when the 8051 is initialized. If a value is pushed onto the stack at the same time, the value will be stored in the internal RAM

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).



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 mean 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