Tishk International University
Mechatronics Engineering Department
Embedded Systems and Real-time
Lecture 3: 24-10-2024

Embedded Systems and Real-time - Microcontroller

Dr. Rand Basil Alhashimie

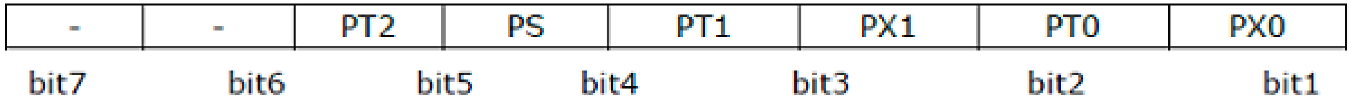
Email: rand.basil@tiu.edu.iq

8051 Microcontroller

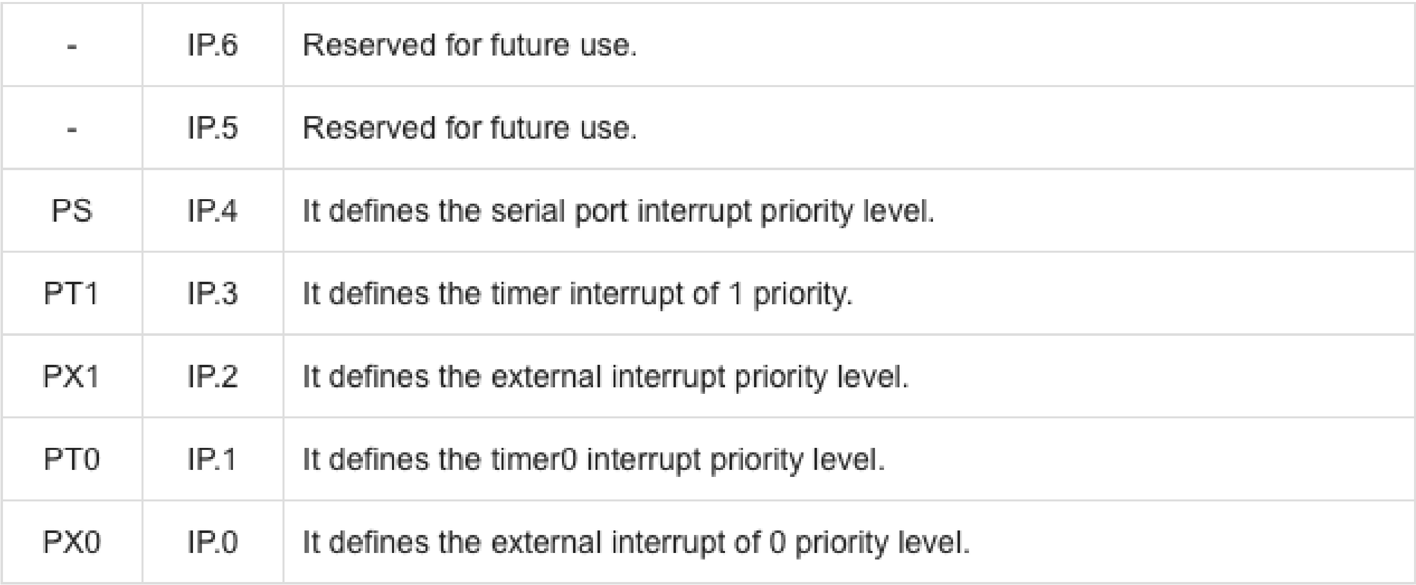
IP (Interrupt Priority) Register

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

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



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

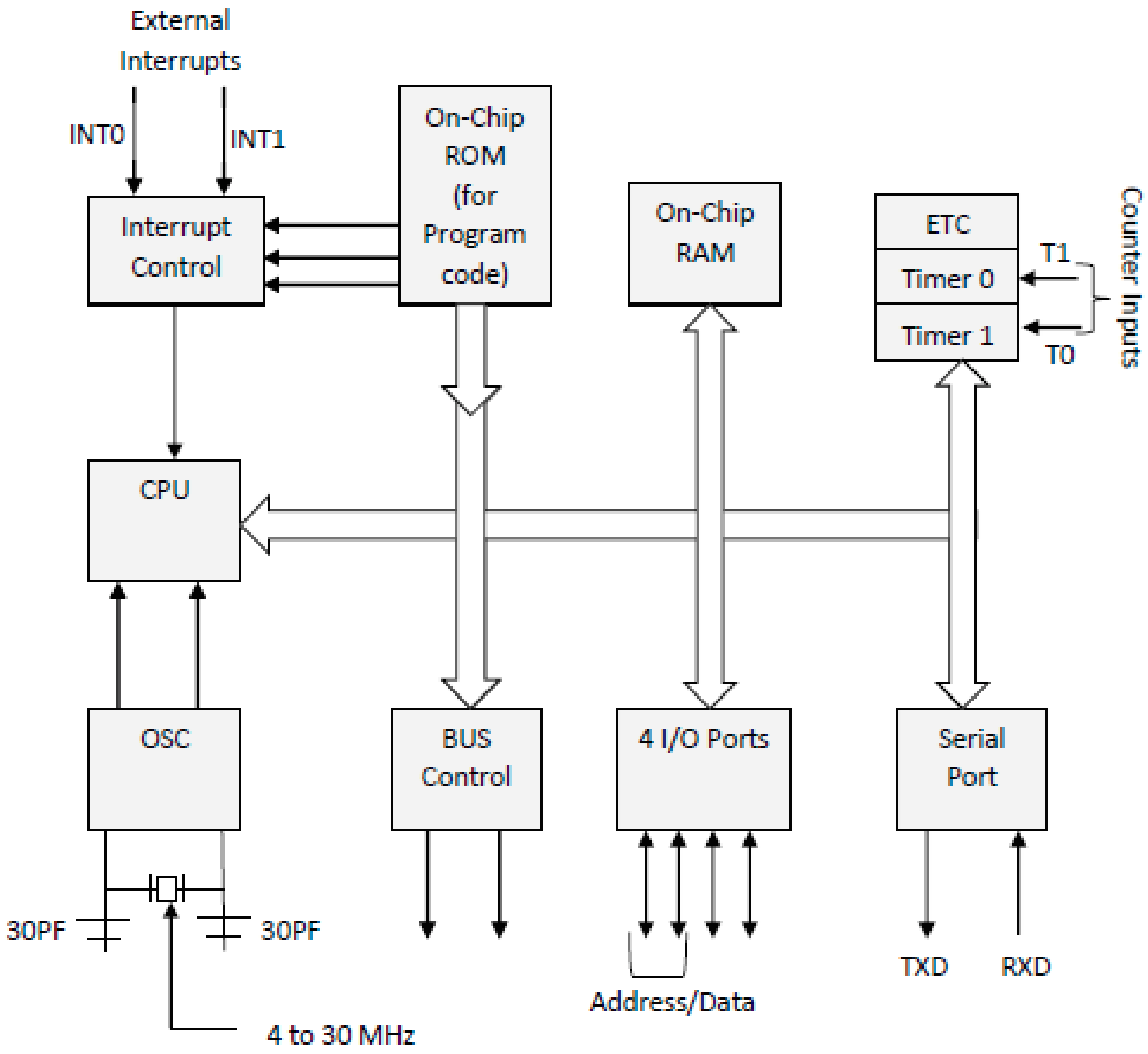


-	IP.6	Reserved for future use.
-	IP.5	Reserved for future use.
PS	IP.4	It defines the serial port interrupt priority level.
PT1	IP.3	It defines the timer interrupt of 1 priority.
PX1	IP.2	It defines the external interrupt priority level.
PT0	IP.1	It defines the timer0 interrupt priority level.
PX0	IP.0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

Internal Structure 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.

8051 employs Harvard architecture. It has some peripherals such as 32 bit digital I/O, Timers and Serial I/O.



8051 Clock and Instruction Cycle

- In 8051, one instruction cycle consists of twelve (12) clock cycles. Instruction cycle is sometimes called as Machine cycle by some authors. In 8051, each instruction cycle has six states (S 1 - S 6). Each state has two pulses (P1 and P2)

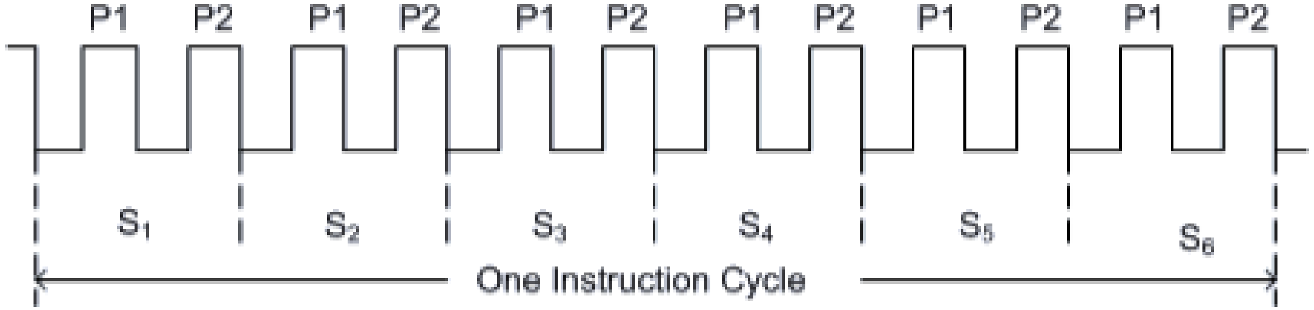


Fig. : Instruction cycle of 8051

8051 Microcontroller: Types of Memory

The 8051 has three very general types of memory. To effectively program the 8051 it is necessary to have a basic understanding of these memory types .

- **On-Chip Memory:** refers to any memory (Code, RAM, or other) that physically exists on the microcontroller itself. On-chip memory can be of several types, but we'll get into that shortly.
- **External Code Memory:** is code (or program) memory that resides off-chip. This is often in the form of an **external EPROM**.
- **External RAM:** is RAM memory that resides off-chip. This is often in the form of standard static RAM or flash RAM.
- **Code Memory:** Code memory is the memory that holds the actual 8051 program that is to be run. This memory is limited to 64K and comes in many shapes and sizes: Code memory may be found on-chip, either ***burned into the microcontroller as ROM or EPROM***. Code may also be stored completely off-chip in an external ROM or, more commonly, an external EPROM. ***Flash RAM*** is also another popular method of storing a program. Various combinations of these memory types may also be used--that is to say, it is possible to have 4K of code memory on-chip and 64k of code memory off-chip in an EPROM. When the program is stored on-chip the 64K maximum is often reduced to 4k, 8k, or 16k. This varies depending on the version of the chip that is being used. Each version offers specific capabilities and one of the distinguishing factors from chip to chip is how much ROM/EPROM space the chip has. However, code memory is most commonly implemented as off-chip EPROM.

External RAM

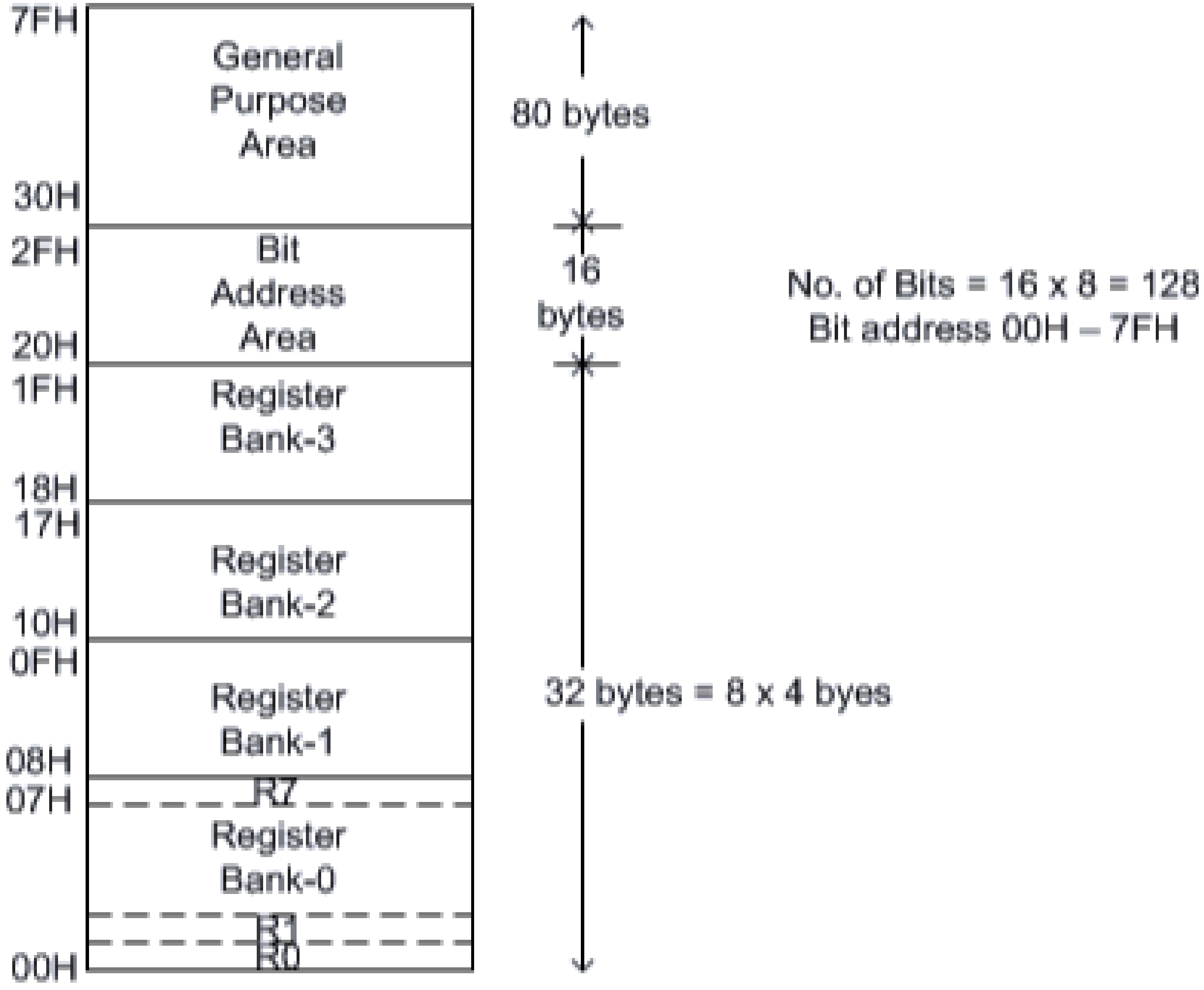
External RAM As an obvious opposite of Internal RAM, the 8051 also supports what is called External RAM. As the name suggests, External RAM is any random access memory which is found off-chip. Since the memory is off-chip it is not as flexible in terms of accessing, and is also slower. For example, to increment an Internal RAM location by 1 requires only 1 instruction and 1 instruction cycle. To increment a 1-byte value stored in External RAM requires 4 instructions and 7 instruction cycles. In this case, external memory is 7 times slower! What External RAM loses in speed and flexibility it gains in quantity. While Internal RAM is limited to 128 bytes the 8051 supports External RAM up to 64K.

On Chip Memory

The 8051 includes a certain amount of on chip memory. On-chip memory is really one of two (SFR) memory. the 8051 has bank of 128 bytes of Internal RAM. This Internal RAM is found on-chip on the 8051 so it is the fastest RAM available, and it is also the most flexible in terms of reading, writing, and modifying it’s contents. Internal RAM is volatile, so when the 8051 is reset this memory is cleared. The 128 bytes of internal ram is subdivided as shown on the memory map. The first 8 bytes (00h- 07h) are "register bank 0". By manipulating certain SFRs, a program may choose to use register banks1, 2, or 3. These alternative register banks are located in internal RAM in addresses 08h through1Fh.The 80 bytes remaining of Internal RAM, from addresses 30h through 7Fh, may be used by user variables that need to be accessed frequently or at high-speed. This area is also utilized by the microcontroller as a storage area for the operating stack. This fact severely limits the 8051’s stack since, as illustrated in the memory map, the area reserved for the stack is only 80 bytes--and usually it is less since this 80 bytes has to be shared between the stack and user variables.

128 bytes of Internal RAM Structure (lower address space)

The lower 32 bytes are divided into 4 separate banks. Each register bank has 8 registers of one byte each. A register bank is selected depending upon two bank select bits in the PSW register. Next 16bytes are bit addressable. In total, 128bits (16X8) are available in bit addressable area. Each bit can be accessed and modified by suitable instructions. The bit addresses are from 00H (LSB of the first byte in 20H) to 7FH (MSB of the last byte in 2FH). Remaining 80bytes of RAM are available for general purpose.



1. Register Banks

The 8051 uses 8 "R" registers which are used in many of its instructions. These "R“ registers are numbered from 0 through 7 (R0, R1,R2, R3, R4, R5, R6, and R7). These registers are generally used to assist in manipulating value sand moving data from one memory location to another. For example, to add the value of R4 to the Accumulator, we would execute the following instruction: ADD A, R4 When the 8051 is first booted up, register bank 0 (addresses 00h through 07h) is used by default. However, your program may instruct the 8051 to use one of the alternate register banks; i.e. register banks 1, 2, or 3. In this case, R4 will no longer be the same as Internal RAM address 04h. For example, if your program instructs the 8051 to use register bank 3, "R" register R4 will now be synonymous with Internal RAM address 1Ch. The concept of register banks adds a great level of flexibility to the 8051, especially when dealing with interrupts However, always remember that the register banks really reside in the first 32 bytes.

2. Bin Memory

The 8051, being a communications oriented microcontroller, gives the user the ability to access a number of bit variables. These variables may be either 1 or 0. There are 128 bit variables available to the user, numbered 00h through 7Fh. The user may make use of these variables with commands such as SETB and CLR. It is important to note that Bit Memory is really a part of Internal RAM. In fact, the 128 bit variables occupy the 16 bytes of Internal RAM from 20h through 2Fh. Thus, if you write the value FFh to Internal RAM address 20h you've effectively set bits 00h through 07h.But since the 8051 provides special instructions to access these 16 bytes of memory on a bit by bit basis it is useful to think of it as a separate type of memory. However, always keep in mind that it is just a subset of Internal RAM—an that operations performed on Internal RAM can change the values of the bit variables. use bit variables, you may use Internal RAM locations Bit variables 00h through 7Fh are for user defined functions in their programs. However, bit variables 80h and above are actually used to access certain SFRs on a bit-by-bit basis.

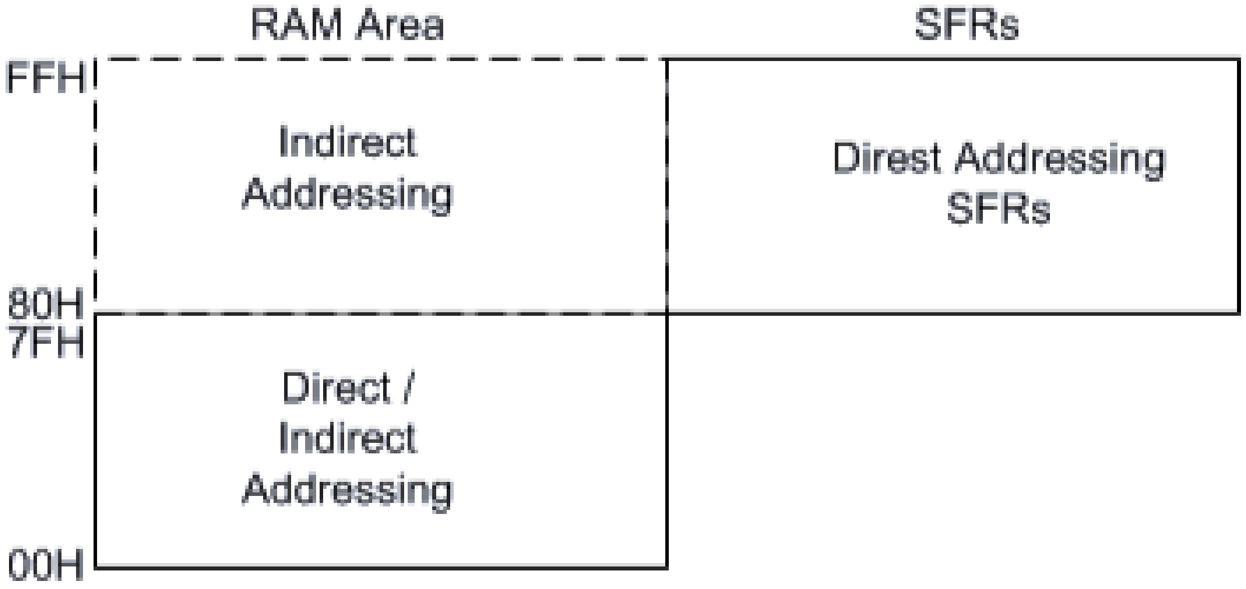
For example, if output lines P0.0 through P0.7 are all clear (0) and you want to turn on the P0.0 output line you may either execute: MOV P0,#01h || SETB 80h. Both these instructions accomplish the same thing. However, using the SETB command will turn on the P0.0 line without effecting the status of any of the other P0 output lines. The MOV command effectively turns off all the other output lines which, in some cases, may not be acceptable.

3. Special Function Register (SFR) Memory

Special Function Registers (SFRs) area of memory that control specific functionality of the 8051 processor. For example, four SFRs permit access to the 8051's 32 input/output lines. Another SFR allows a program to read or write to the 8051's serial port. Other SFRs allow the use to set the serial baud rate, control and access timers, and configure the 8051's interrupt system. When programming, SFRs have the illusion of being Internal Memory. For example, if you want to write the value "1" to Internal RAM location 50 hex you would execute the instruction: MOV 50h,#01Similarly, if you want to write the value "1" to the 8051's serial port you would write this value to the SBUF SFR, which has an SFR address of 99 Hex. Thus, to write the value "1" to the serial port you would execute the instruction: MOV 99h,#01h As you can see, it appears that the SFR is part of Internal Memory. This is not the case. When using this method of memory access (it's called direct address), any instruction that has an address of 00h through 7Fh refers to an Internal RAM memory address; any instruction with an address of 80h through FFh refers to an SFR control register.

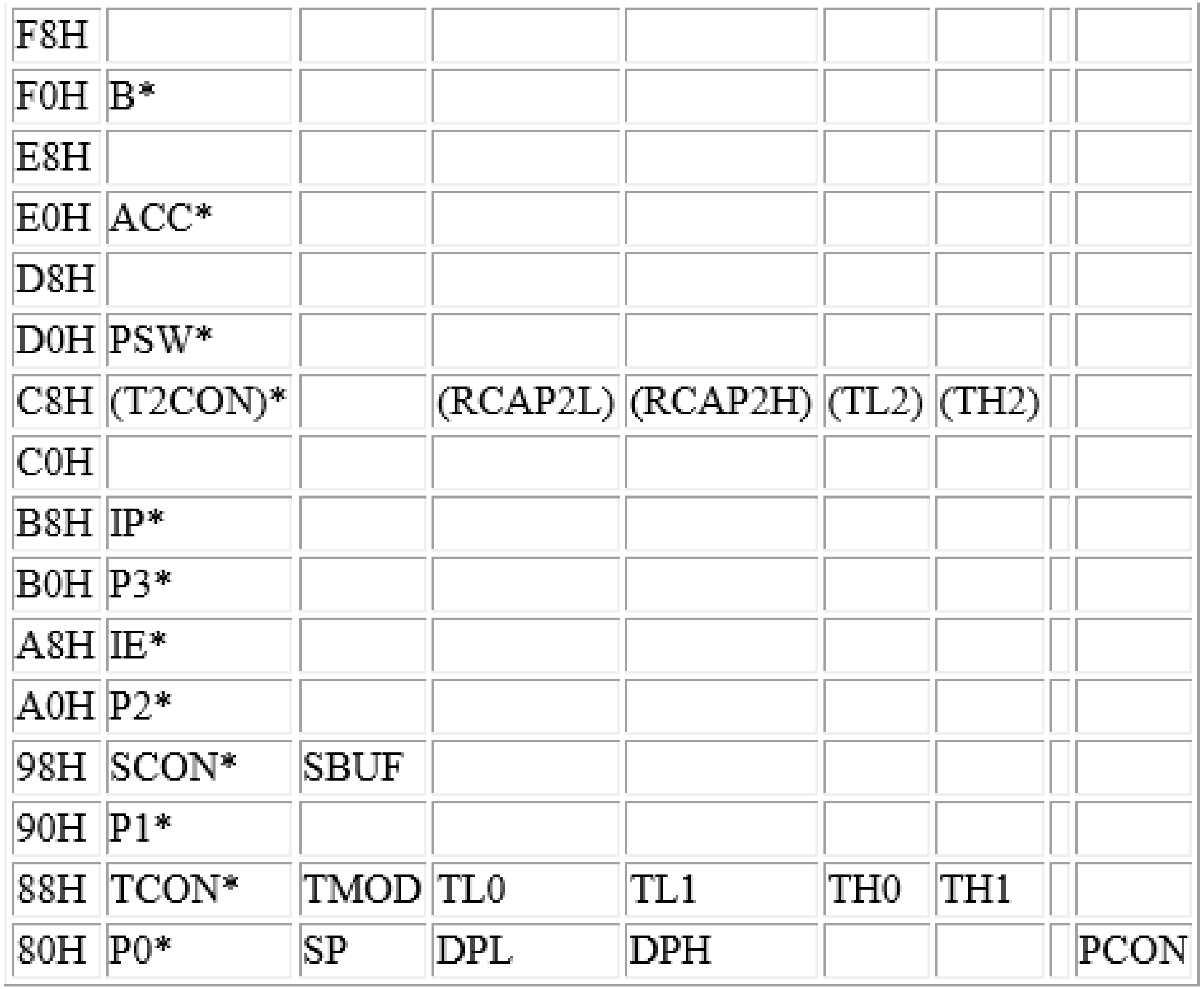
Special Function Register (SFR) Memory

The special function registers (SFRs) are mapped in the upper 128 bytes of internal data memory address. Hence there is an address overlap between the upper 128 bytes of data RAM and SFRs. Please note that the upper 128 bytes of data RAM are present only in the 8052 family. The lower128 bytes of RAM (00H - 7FH) can be accessed both by direct or indirect addressing while the upper 128 bytes of RAM (80H - FFH) are accessed by indirect addressing. The SFRs (80H - FFH) are accessed by direct addressing only. This feature distinguishes the upper 128 bytes of memory from the SFRs, as shown in fig.



Special Function Register (SFR) Memory

The set of Special Function Registers (SFRs) contains important registers such as Accumulator, Register B, I/O Port latch registers, Stack pointer, Data Pointer, Processor Status Word (PSW) and various control registers. Some of these registers are bit addressable (they are marked with a* in the diagram below). The detailed map of various registers is shown in the following figure. Address



F8H								
F0H	B*							
E8H								
E0H	ACC*							
D8H								
D0H	PSW*							
C8H	(T2CON)*		(RCAP2L)	(RCAP2H)	(TL2)	(TH2)		
C0H								
B8H	IP*							
B0H	P3*							
A8H	IE*							
A0H	P2*							
98H	SCON*	SBUF						
90H	P1*							
88H	TCON*	TMOD	TL0	TL1	TH0	TH1		
80H	P0*	SP	DPL	DPH				PCON

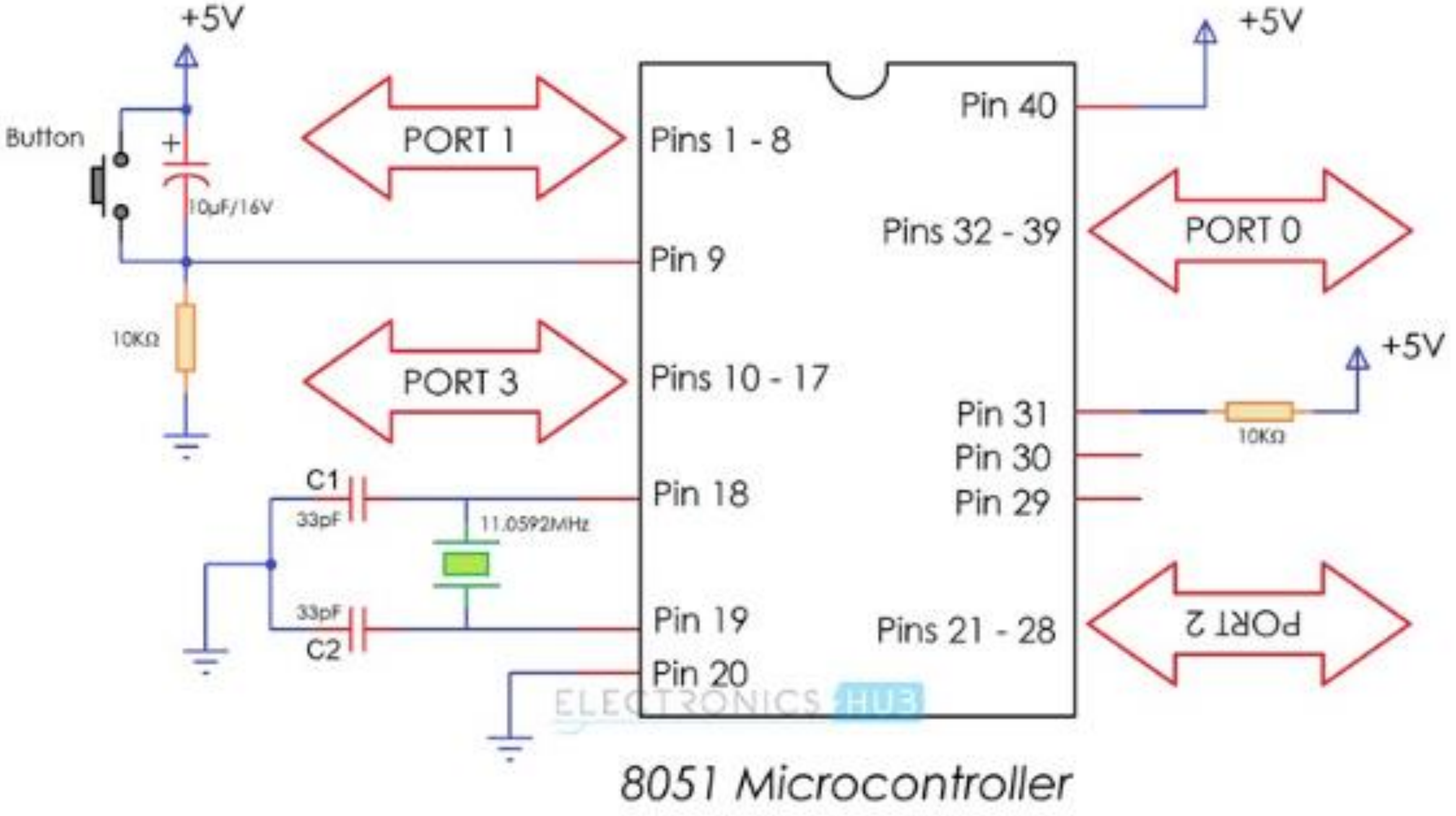
8051 Microcontroller Function Registers

There are many ways to categorize these 21 Special Function Registers but I find the following way as an appropriate one. The 21 Special Function Registers of 8051 Microcontroller are categorized in to seven groups. They are:

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

8051 Microcontroller Basic Circuit

Now that we have seen the 8051 Microcontroller Pin Diagram and corresponding Pin Description, we will proceed to the basic circuit or schematic of the 8051 Microcontroller. The following image shows the basic circuit of the 8051 Microcontroller.



8051 Microcontroller Basic Circuit

- This basic circuit of 8051 microcontroller is the minimal interface required for it to work. The basic circuit includes a Reset Circuit, the oscillator circuit and power supply. Let us discuss a little bit deeper about this basic circuit of 8051 Microcontroller.
- First is the power supply. Pins 40 and 20 (VCC and GND) of the 8051 Microcontroller are connected to +5V and GND respectively.
- Next is the Reset Circuit. A logic HIGH (+5V) on Reset Pin for a minimum of two machine cycles (24 clock cycles) will reset the 8051 Microcontroller. The reset circuit of the 8051 Microcontroller consists of a capacitor, a resistor and a push button and this type of reset circuit provides a Manual Reset Option. If you remove the push button, then the reset circuit becomes a Power-On Reset Circuit.

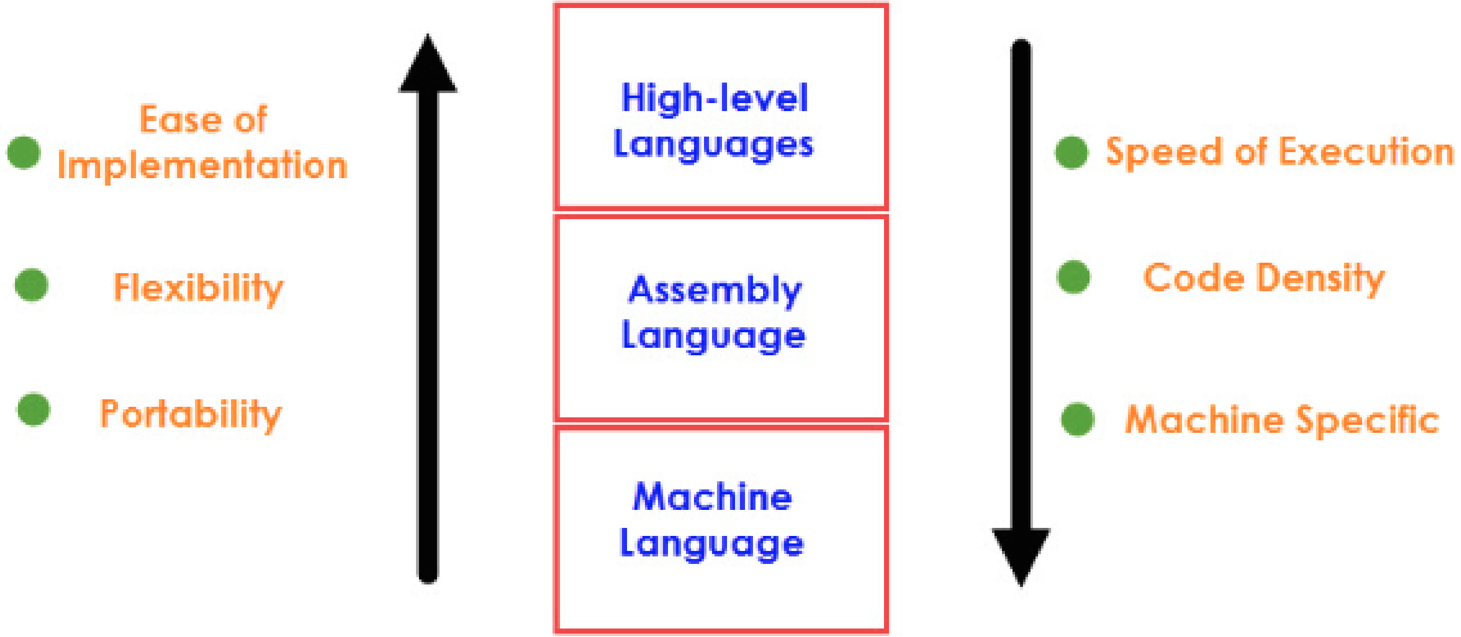
8051 Microcontroller Basic Circuit

- The next part of the basic circuit of the 8051 Microcontroller is the Oscillator Circuit or the Clock Circuit. A Quartz Crystal Oscillator is connected across XTAL1 and XTAL2 pins i.e. Pins 19 and 18. The capacitors C1 and C2 can be selected in the range of 20pF to 40pF.
- As mentioned in the 8051 Microcontroller Pin Description, PORTS 1, 2 and 3, all have internal pull – ups and hence can be directly used as Bidirectional I/O Ports. But, we need to add external Pull – ups for PORT 0 Pins in order to use it as an I/O Port.
- Generally, a 1KΩ Resistor Pack of 8 Resistors is used as a Pull – up for the PORT 0 of the 8051 Microcontroller.
- In this tutorial, we have seen about the 8051 Microcontroller Pin Diagram, Pin Description and the Basic Circuit of 8051 Microcontroller. In the next tutorial, we will continue with the architecture and few other features of 8051 Microcontroller.

8051 Microcontroller Programming Languages

The three levels of Programming Languages are:

- Machine Language
- Assembly Language
- High-level Language



8051 Microcontroller Programming Languages

Machine language

- In Machine language or Machine Code, the instructions are written in binary bit patterns i.e. combination of binary digits 1 and 0, which are stored as HIGH and LOW Voltage Levels. This is the lowest level of programming languages and is the language that a Microcontroller or Microprocessor actually understands.

Assembly Language

- The next level of Programming Language is the Assembly Language. Since Machine Language or Code involves all the instructions in 1’s and 0’s, it is very difficult for humans to program using it.
- Assembly Language is a pseudo-English representation of the Machine Language. The 8051 Microcontroller Assembly Language is a combination of English like words called Mnemonics and Hexadecimal codes.
- It is also a low level language and requires extensive understanding of the architecture of the Microcontroller.

High-level Language

- The name High-level language means that you need not worry about the architecture or other internal details of a microcontroller and they use words and statements that are easily understood by humans.
- Few examples of High-level Languages are BASIC, C Pascal, C++ and Java. A program called Compiler will convert the Programs written in High-level languages to Machine Code.

NEXT LECTURE

- PIC Microcontroller
  - Define, types and structure.