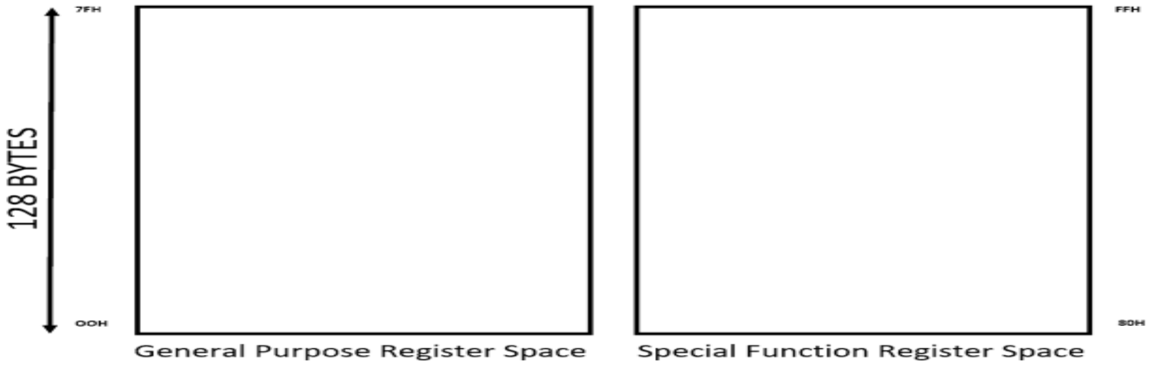
SPECIAL FUNCTION REGISTERS OF THE 8051 MICROCONTROLLER.

These registers, abbreviated as SFRs, control various operations of the microcontroller

Where are the Special Function Registers located?

If you are familiar with the architecture of the 8051 microcontroller, you might remember that it has 128 bytes of RAM. Well, that is not entirely true. The 8051 microcontroller has 256 bytes of RAM divided into two parts; a 128-byte general-purpose area and a 128-byte special function register space.


128 BYTES
7FH
00H
FFH
80H
General Purpose Register Space
Special Function Register Space

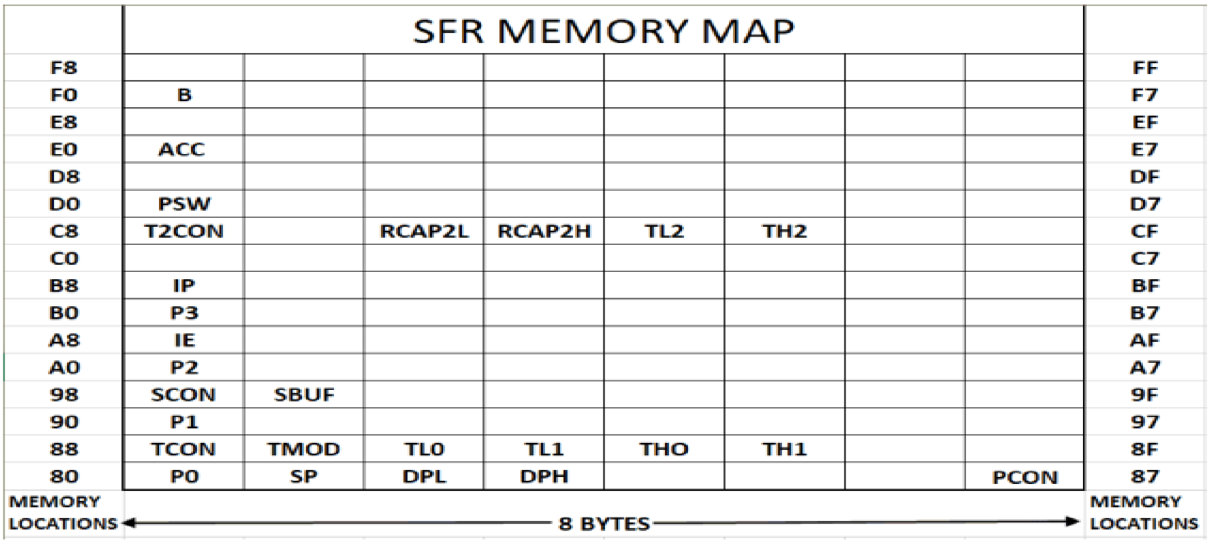
RAM memory partition in 8051

The general-purpose register space stores all the data that generates during the execution of a program. In contrast, the special function register area maintains a set of special function registers that are responsible for managing the various operations occurring in the microcontroller.

The SFR space (80H to FFH) can be accessed by the programmer using direct addressing to configure them to perform specific tasks, but storing normal program data in this RAM space is not possible.

Therefore, the 8051 microcontroller has an effective general-purpose RAM space of 128 bytes (00H to 7FH).

Note: General-purpose registers can be accessed using both indirect and direct addressing methods, but Special function registers can be accessed using direct addressing only.



SFR MEMORY MAP

F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	T2CON		RCAP2L	RCAP2H	TL2	TH2			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	THO	TH1			8F
80	P0	SP	DPL	DPH				PCON	87
MEMORY LOCATIONS	← 8 BYTES →								MEMORY LOCATIONS

Looking at the picture above, you might be wondering, can one place data in the empty spaces? Well, it is not possible to write data in the SFR space even if some places are empty as they are kept free by the manufacturer to add functionalities to newer versions of the same microcontrollers. Now that you know about this hidden memory space in the RAM let us talk about the various registers that comprise the SFR space.

Control Special Purpose Registers

Control parameters in the 8051 microcontroller for Timers, Serial communication, and interrupts use Special registers known as special-purpose registers.

Power control (PCON)

The PCON register is responsible for managing the power modes of the 8051 microcontrollers. It is placed at the address 87H and is not bit addressable.

In most embedded system devices, power management is the biggest constraint for any developer. The 8051 offers a power-down mode and an idle mode to decrease its power requirements. In the power-down mode, the clock of the microcontroller is frozen, stopping all the operations of the microcontroller. This drops the voltage requirement for the working of the microcontroller from 5v to 2v, and the current demand reduces below 60 microamperes.

Power down mode

During the power-down mode, the status of all the SFRs and input/output ports remains the same, and the only way to wake up this sleepy microcontroller is by using a hardware reset. When the microcontroller comes back to its normal mode after the reset, the values of all the SFRs are lost, but the data in the general-purpose RAM remains the same.

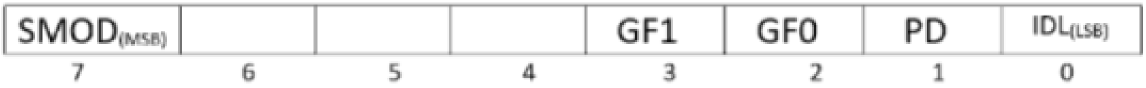
Note: The reset signal should be long enough (10 msec) so that the oscillator can start functioning normally. Also, Vcc should not be dropped below 5v until the controller is in power-down mode.

Idle mode

In the case of the idle mode, the most power-hungry unit, the CPU of the microcontroller, is turned off. All the other peripherals like the timer, serial communication, and interrupts keep functioning normally. Also, the statuses of the CPU, Accumulator, and the stack pointer remain as it is.

There are two ways to pull the 8051 microcontroller out from the idle mode. An interrupt that changes the value at PCON.0. Or a hardware reset.

Here's what the innards of the PCON register contain.



SMOD(MSB)				GF1	GF0	PD	IDL(LSB)
7	6	5	4	3	2	1	0

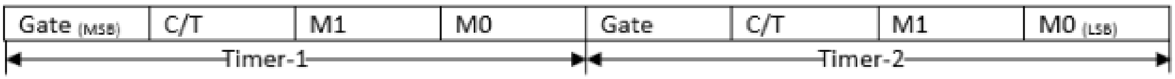
PCON REGISTER AT 87H

- **SMOD:** The baud rate in serial communication doubles when this bit is set to 1.
- **GF1 and GF0:** are general-purpose flag bits.
- **Power Down(PD)**: When this bit is set to 1, the microcontroller goes into power-down mode.
- **Idle(IDL):** This bit puts the microcontroller in idle mode when the bit is 1.
- The microcontroller does not use the rest of the three bits.

Timer mode (TMOD)

TMOD stands for Timer Mode, and as the name suggests, it is responsible for setting the mode of a timer among other things. It is an 8-bit register that is placed at the address 89H in the ROM space and is not bit addressable. What this means is that each value in the 8-bit register can't be changed independently.

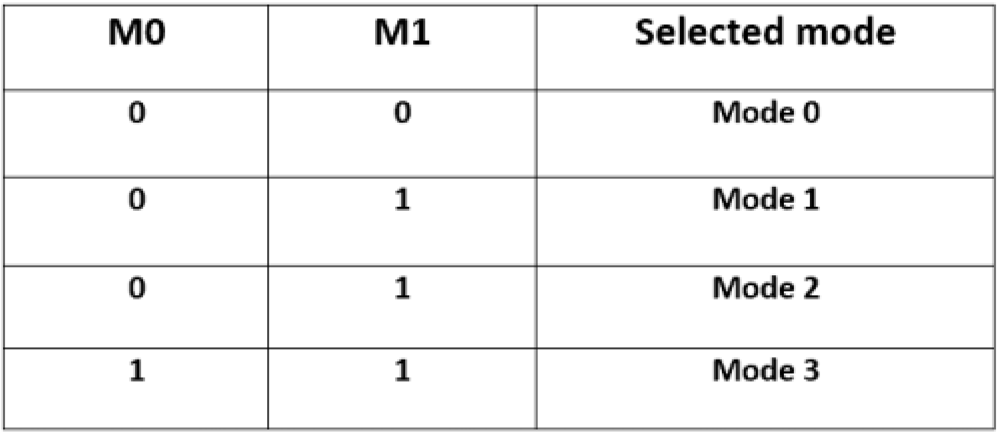
This 8-bit register is divided into two 4-bit registers to manage T0 and T1, respectively. The lower 4 bits manage the modes for T0, and the higher four manage T1. These four bits are Gate, C/T, M1 and M0



Gate (MSB)	C/T	M1	M0	Gate	C/T	M1	M0 (LSB)
Timer-1				Timer-2			

STRUCTURE OF TMOD REGISTER-ADDRESS 89H

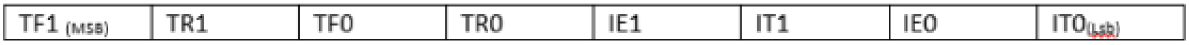
- **Gate:** Most microcontrollers need a way to start a timer. In the case of the 8051 microcontrollers, there are two ways to start the timers; by external hardware or software. The Gate register in TMOD is used to decide how the timer begins. If it is 1, then the timer will start using hardware (external pulse from port 3). If it is set to 0, it will start by software mechanism when it receives a control signal one from the TR1 in the TCON register.
- **C/T:** These timers can be used as both timers and counters to keep time and to count external events. To use it as a timer, the C/T bit is set to 0, whereas it is set to 1 if it needs to be used as a counter of events.
- **M0 and M1:** there are four modes that 8051 offers. This means that we need two bits to set these modes.



M0	M1	Selected mode
0	0	Mode 0
0	1	Mode 1
0	1	Mode 2
1	1	Mode 3

Timer control (TCON)

Now that we have got TMOD out of the way let’s go to the next control SFR that is used by timers, TCON. TCON stands for timer control and is used to send the control signals for the functioning of the timer. The TCON register is bit addressable and is placed at the address 88H in the ROM. It is an 8-bit register which starts the timer and also contains the flag which gets updated when the timer overflows.



TF1 (MSB)	TR1	TF0	TR0	IE1	IT1	IE0	IT0(Lsb)

STRUCTURE OF TCON REGISTER IN RAM-ADDRESS 88H

TF1 and TF0 are timer flags that indicate overflow condition in timers 1 and timer 0, respectively. When an overflow condition is reached, the value is changed to 1 from 0. TR0 and TR1 are used to start timers 0 and 1. When a value 1 loads into these registers, the timers start counting from the values which are inside them.

You might be wondering what is this overflow? Well, it is just like your water tank reaching its complete capacity and overflowing. The 8051 has two 16 bit timers which count up from 0000H to FFFFH. Once the timer reaches a value of FFFFH it has no more values it can count up to making it overflow, but unlike spilling water in the case of your water tank the microcontroller resets the value to 0000H and sets the TF1/TR0 bit to 1 telling the programmer that the timer has overflowed.

Let’s say your microcontroller is busy performing some mathematical operations, and in the middle of that, some actuator needs to be serviced. In such cases, the external interrupts come in handy. An interrupt, in essence, is a way of stoping the microcontrollers’ current operations and making it perform some other task due to higher importance.

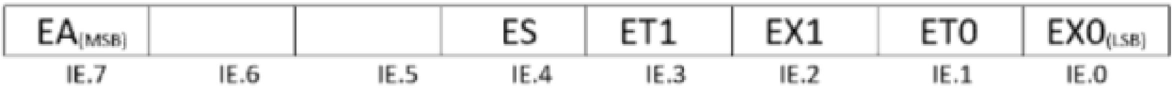
The 8051 microcontroller has two external interrupt ports P3.2 and P3.3 to manage external devices of high priority. To alert the microcontroller, peripheral devices use certain signals. It can either be an edge-triggered interrupt or a low-level signal which triggers the interrupt. IT1(P3.3) and IT0(P3.2) are used to select which kind of signal will be used to interrupt the operations of the microcontroller. If IT1 is set to 1, then a falling edge signal will be seen as the interrupt, whereas if it is 0, a low-level signal for a certain duration will be used to interrupt the microcontroller.

IE0 and IE1 are used to tackle the problem of looping inside interrupts, which occurs during external edge-triggered interrupts. The CPU sets IE1(P3.3)/IE0(P3.2) as 1 when it encounters an edge-triggered signal at any of the two ports. When this bit is 1, the CPU ignores any other falling edge signal on the port, preventing reoccurring interrupts.

Interrupt Enable(IE)

We have talked a bit about interrupts in the previous section, but to make them clear, let’s use an analogy. Let us say you are sitting and binge-watching your favorite TV show, and in the middle of your favorite scene, you get a phone call. Now you have to pause the show and attend the call. The interrupt is like a phone call for the CPU. It forces it to pause all its work and tend to the interrupt (phone call). Using interrupts can help you to optimize your program by efficiently using CPU resources.

The interrupt enable register is used to manage the six interrupts the 8051 microcontroller has. By default, all the interrupts are turned off and need to be turned on by the programmer. The interrupt enable (IE) register is a bit addressable register and is placed at the A8H address.



EA$_{(MSB)}$			ES	ET1	EX1	ET0	EX0$_{(LSB)}$
IE.7	IE.6	IE.5	IE.4	IE.3	IE.2	IE.1	IE.0

INTERRUPT ENABLE REGISTER AT A8H

- To enable any interrupt in the 8051 microcontroller, EA bit needs to be set to 1. By default, it is set to 0, and no interrupts are enabled.
- **ES:** bit enables serial communication interrupt when 1 is inserted into it and is disabled when the value is 0
- **ET1:** enables the interrupt for timer 1 when it is set to 1 and is disabled when it is 0
- **EX1:** this bit is used to enable external interrupt 1(p3.3)
- **ET0:** this bit is used to enable the interrupt for timer 0
- **EX0:** it is used to enable the external interrupt 0 port(p3.2)
- The rest of the bits can’t be used by the programmer

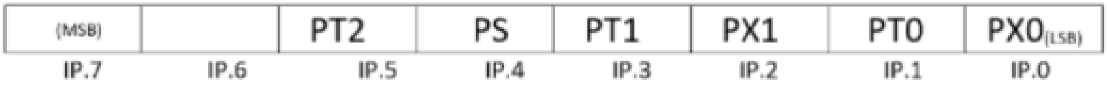
Interrupt Priority(IP)

The interrupts in the 8051 microcontrollers have certain priorities assigned to them. This helps the microcontroller to decide which interrupt to service before if two of them occur at the same time. The priorities of the interrupts are as follows:

1. External interrupt 0
2. Timer interrupt 0
3. External interrupt 1
4. Timer interrupt 1
5. Serial interrupt

To change the priorities of these interrupts, the Interrupt Priority register is used.

Note: If two bits in the IP register are raised, then the priorities mentioned above are used to make the decision on which interrupt will be serviced before. If any bit in the IP register is set to 1, then the priority of that interrupt increases.



$_{(MSB)}$		PT2	PS	PT1	PX1	PT0	PX0$_{(LSB)}$
IP.7	IP.6	IP.5	IP.4	IP.3	IP.2	IP.1	IP.0

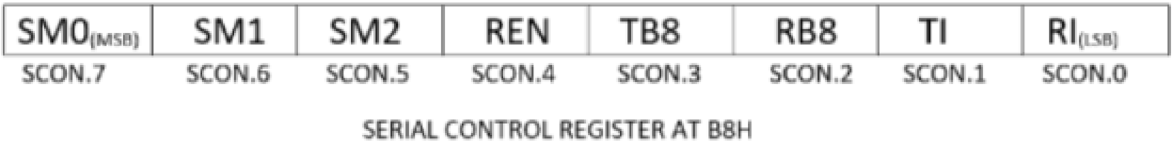
INTERRUPT PRIORITY REGISTER AT B8H

If any of the bits in the IP register has a value one its priority of that interrupt is increased

- **PT2:** Increases the priority of timer 2 (for 8052 only)
- **PS:** Increases the priority of the serial interrupt
- **PT1:** Increases the priority of timer 1
- **PX1:** Increases the priority of external interrupt 1
- **PT0:** Increases the priority of timer 0
- **PX1:** Increases the priority of external interrupt 0

Serial control(SCON)

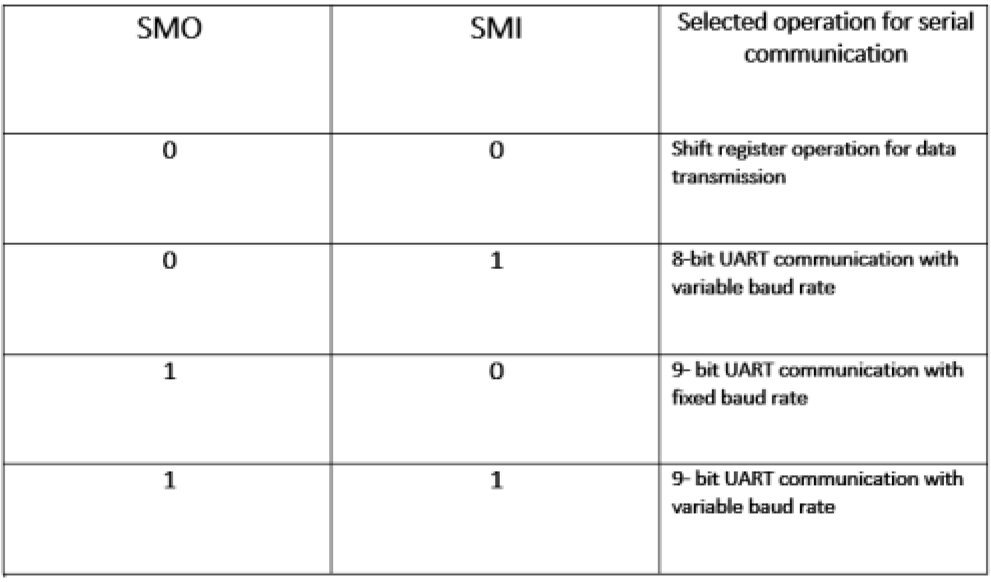
The 8051 microcontroller can communicate with peripherals using serial communication. To manage this asynchronous communication between devices, the SCON register is used. The SCON register is placed at the address 98H and is bit addressable.



SM0(MSB)	SM1	SM2	REN	TB8	RB8	TI	RI(LSB)
SCON.7	SCON.6	SCON.5	SCON.4	SCON.3	SCON.2	SCON.1	SCON.0

SERIAL CONTROL REGISTER AT B8H

- **SM2:** This bit is used to enable multiprocessor communication in modes 2 and 3.
- **REN:** The Ren bit is used to enable or disable reception.
- **TB8:** 9th bit is stored in this register during mode 2 and 3 while transmitting data
- **RB8:** 9th bit is stored in this register during mode 2 and 3 while receiving data
- **TI:** This interrupt is raised by the software when the transmission is complete for a block of data.
- **RI:** This interrupt is raised by the software when a complete block of data is received.
- **SMO and SM1**



SMO	SMI	Selected operation for serial communication
0	0	Shift register operation for data transmission
0	1	8-bit UART communication with variable baud rate
1	0	9- bit UART communication with fixed baud rate
1	1	9- bit UART communication with variable baud rate

We’ll study serial communication in 8051 in detail at a later point in this 8051 course.

Input-output related SFRs

A microcontroller would be rendered useless if it could not communicate with the various sensors and actuators in its environment. The I/O related SFRs help the microcontroller manage these ports efficiently.

Port SFRs

The 8051 microcontroller has four I/O ports, all of which have eight pins each. To manage these ports, there are four SFR Registers, which are bit addressable. The main job of these 8 bit SFRs is to set the direction of data flow from each pin. If a bit is set to 1, then the pin acts as an input port. If a bit is set to 0, then the port acts as an output port. You can read all about the I/O ports of the 8051 here.

Other SFRs

Apart from the above-mentioned registers, there are a few other registers which help mage calculations and addressing in the 8051 microcontroller they are

Accumulator

The accumulator is a special register that connects to the ALU, and all the results of any computation performed in the CPU are stored there. It is a bit addressable register and is placed at E0H. There is another register know as the B register, which stores the results when the data is too large and cannot be stored in an 8-bit register. This happens during operations like multiplication and division. It is placed at F0H and is bit addressable like the accumulator.

Stack pointer

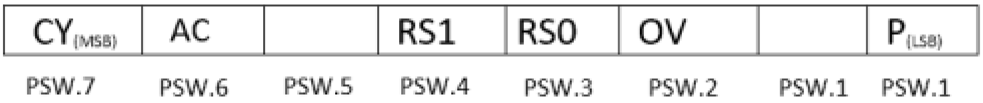
- A set of registers in the RAM form the stack.
- This stack of registers is used to store either data or addresses which are required during the execution of a program.
- To access the data in the stack, a special 8-bit register known as the stack pointer is used.
- When the microcontroller is powered up, the stack pointer points to the location 07H and grows upwards till the memory location 1FH.
- This provides a total of 24 bytes for the stack area.
- It uses special push and pop instructions to place and retrieve data stored in the stack, respectively.
- If more memory space for the stack is required, it can be moved to the scratch-pad in the location 30H to 7FH in the general purpose RAM.

All of this might be a bit overwhelming for you at the moment, but don’t worry, we will be covering the entire general-purpose RAM space in the next article, which will make things crystal clear.

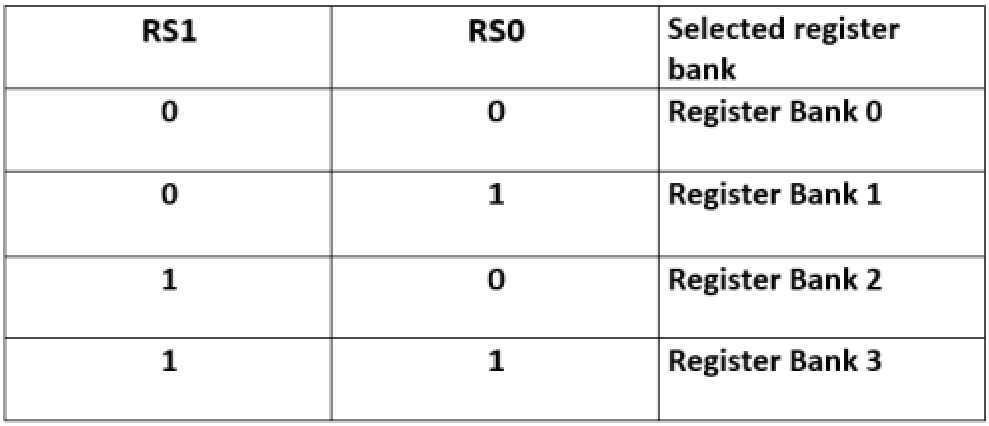
Program status word (PSW)

It is a special function register (SFR) that is responsible for managing the various mathematical overheads like carries, auxiliary carries, and much more. Although the PSW register is 8-bit wide, only 6 bits are used by the 8051 microcontrollers. The other two bits are user-definable and can be used according to the needs of the programmer.

The PSW register comprises four conditional flags, namely Carry (CY), auxiliary carry (AC), Parity (P), and Overflow (OV). Two bits in the PSW register are used to select the register bank in the RAM for storing data. By default, register bank 0 stores all the data. Note that the 8051 has four register banks containing eight registers each. The PSW register is bit addressable, which allows the programmer to access each bit individually.



CY$_{(MSB)}$	AC		RS1	RS0	OV		P$_{(LSB)}$
PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.1



RS1	RS0	Selected register bank
0	0	Register Bank 0
0	1	Register Bank 1
1	0	Register Bank 2
1	1	Register Bank 3

- The 8051 microcontroller has an 8-bit architecture. Sometimes during mathematical operations, the ALU produces results greater than 8 bits. To show the programmer that the result is greater than 8 bits, the microcontroller changes the CY bit from 0 to 1.
- The Auxilary Carry is another carry flag which helps in performing BCD addition. This flag is raised when there is a carry from D3 to D4.
- Both PSW.5 and PSW.1 can be configured according to the needs of the programmer.
- RS1 and RS0 are used to select a particular register bank in the RAM space.
- The overflow flag is raised from 0 to 1 if there is an error in any mathematical calculation. Sometimes during subtraction operation, the result of the calculation is wrong, and to tell the programmer that the result is not correct, this flag is updated.
- The parity flag is used as an even party for the accumulator. This bit is 1 if the number of 1's in the accumulator is odd and 0 otherwise

DPTR

DPTR or the data pointer register, as the name suggests, is a register that points to some data. In any computer system, data is kept at a particular address, and the DPTR is responsible for storing the address. Due to the 16-bit address bus of the 8051 microcontroller, it has a 16-bit DPTR so that it can access all the data locations. It is divided into two parts, namely DPH(4 upper bits) and DPL(4 lower bits). It is not a bit addressable and is placed at locations 82H(DPL) and 83H(DPH)

Serial Buffer (SBUF)

The SBUF register is an intermediate 8-bit buffer which stores the data before it is transmitted or received from any peripheral. This buffer helps to reduce the errors during serial communication between two devices.

Well, we all hope that we had control mechanisms like the 8051 microcontrollers so that we went to the gym every day, but unfortunately, the only thing we have is self-control.