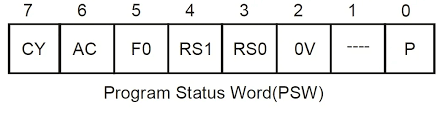
**1. Explain PSW of 8051 Microcontroller.**

The Program Status Word (PSW) is a special function register in the Intel 8051 microcontroller that contains various flags and control bits that reflect the current state of the processor. The PSW is 8 bits wide and is located at address 0xD0 in the 8051's memory-mapped I/O space.



The PSW in the 8051 microcontroller is organized as follows:

- Bit 7 (CY): Carry Flag - This flag is set when an arithmetic operation generates a carry out of the most significant bit (bit 7). It is also used in certain logical operations.

- Bit 6 (AC): Auxiliary Carry Flag - This flag is used for BCD (Binary Coded Decimal) arithmetic. It is set when there is a carry out of bit 3 to bit 4 in an operation.

- Bit 5 (F0): User-Defined Flag 0 - This flag is not used by the 8051 core but can be used by the programmer for any purpose.

- Bit 4 (RS1): Register Bank Select 1 - These two bits, along with RS0 (bit 3 of the PSW), are used to select one of the four register banks in the 8051.

- Bit 3 (RS0): Register Bank Select 0 - See RS1.

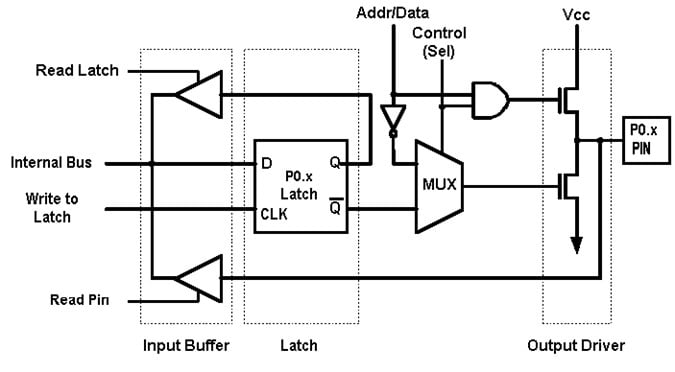
- Bit 2 (OV): Overflow Flag - This flag is set when an arithmetic operation generates an overflow, indicating that the result is too large to be represented in the available number of bits.

- Bit 1 (UD): User-Defined Flag 1 - Similar to F0, this flag is not used by the 8051 core but can be used by the programmer for any purpose.

- Bit 0 (P): Parity Flag - This flag is set if the result of an operation contains an even number of set bits. It is used for parity checking.

The PSW is automatically pushed onto the stack during certain operations, such as interrupts and subroutine calls, and restored when the operation completes. It is also used by conditional branch instructions to determine whether to take a branch based on the state of the flags.

**2. With neatly labelled circuit diagram explain Port 0 & Port 1 structure of 8051.**

Port 0 (P0):

- Port 0 is an 8-bit bidirectional I/O port (P0.0 - P0.7).

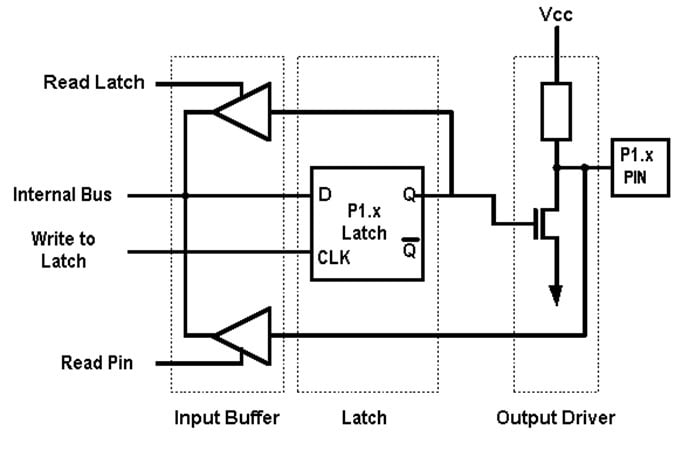
- It is multiplexed with the least significant byte of the address bus (AD0-AD7) in the 8051.

- When used as an output port, each pin can sink up to 20mA.

- When used as an input port, the pins are pulled high by internal pull-up resistors, but can be pulled low externally.

- Port 0 pins can be used for various purposes such as interfacing with external devices like LEDs, switches, sensors, etc.

Port 1 (P1):



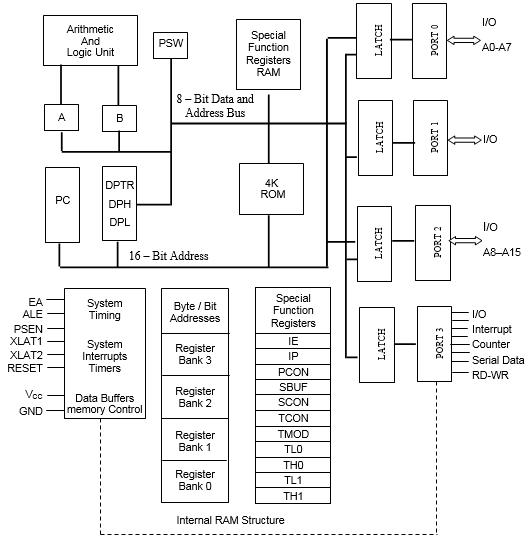
- Port 1 is also an 8-bit bidirectional I/O port (P1.0 - P1.7).

- It is not used for any alternate functions like Port 0.

- Each pin can sink up to 20mA when used as an output.

- When used as an input, the pins are pulled high by internal pull-up resistors and can be pulled low externally.

- Port 1 pins can be used for general purpose I/O like Port 0, but since it does not have alternate functions, it is often used for general purpose I/O where no special function is required.

**3. Explain the architecture of 8051 Microcontroller with a neat diagram.**

The architecture of the 8051 microcontroller is based on a Harvard architecture, which means it has separate memory spaces for program and data. The 8051 architecture consists of several key components:

1. CPU (Central Processing Unit): The CPU of the 8051 is an 8-bit processor that can operate at various clock speeds. It has a set of registers, including the accumulator (A), 8 general-purpose registers (R0-R7), a program counter (PC), a data pointer (DPTR), and various special function registers (SFRs) for controlling and configuring the microcontroller.

2. Memory:

- Program Memory: The 8051 has a 4 KB ROM (Read-Only Memory) for storing the program code.

-Data Memory: It has 128 bytes of RAM (Random Access Memory) for storing data and variables. This RAM is divided into two banks of 128 bytes each, and the current bank is selected using the register bank select bits in the PSW (Program Status Word) register.

3. I/O Ports:

- The 8051 has four 8-bit I/O ports (P0, P1, P2, P3) that can be used for interfacing with external devices.

- Each port can be configured as input or output, and the individual pins can be controlled using software.

4. Timers/Counters:

- The 8051 has two 16-bit timers/counters (T0 and T1) that can be used for various timing and counting applications.

- These timers can be configured to run in different modes, including timer, counter, and baud rate generator modes.

5. Serial Communication:

- The 8051 has a built-in UART (Universal Asynchronous Receiver Transmitter) for serial communication.

- It supports full-duplex communication and can be used for interfacing with other serial devices.

6. Interrupts:

- The 8051 supports both hardware and software interrupts.

- It has five interrupt sources, and each interrupt can be individually enabled or disabled using interrupt enable bits.

7. Clock Circuitry:

- The 8051 requires an external crystal oscillator for providing the clock signal.

- It can operate at frequencies ranging from a few kilohertz to tens of megahertz, depending on the specific model.

8. Memory Organization:

- The 8051 has a total addressable memory space of 64 KB.

- It uses a separate memory space for program memory (ROM) and data memory (RAM).

- The program memory is 4 KB in size and is used to store the program code.

- The data memory is 128 bytes in size and is used to store data and variables. It is divided into two banks of 128 bytes each, and the current bank is selected using the register bank select bits in the PSW register.

9. Memory Addressing:

- The 8051 uses 8-bit addressing for both program and data memory.

- It supports direct addressing, indirect addressing, and indexed addressing modes.

- Direct addressing is used to access memory locations directly.

- Indirect addressing is used with the data pointer (DPTR) register to access data stored at a memory location pointed to by DPTR.

- Indexed addressing is used to access memory locations using an index register (R0 or R1).

10. Stack:

- The 8051 has a hardware stack that is used for storing return addresses during subroutine calls and for storing context during interrupts.

- The stack grows from higher memory addresses to lower memory addresses.

11. Register Bank:

- The 8051 has four register banks, each containing eight registers (R0-R7).

- The current register bank is selected using the register bank select bits in the PSW register.

12. PSW (Program Status Word) Register:

- The PSW register is an 8-bit register that contains various flags and control bits.

- Flags include carry flag (CY), auxiliary carry flag (AC), register bank select bits (RS1 and RS0), overflow flag (OV), user-defined flags (F0 and F1), and parity flag (P).

- The PSW register is used to control the behavior of the microcontroller and to store status information.

13. Data Types:

- The 8051 supports 8-bit, 16-bit, and 32-bit data types.

- It has instructions for handling arithmetic, logic, and bit manipulation operations on these data types.

**4. Explain the various addressing modes of 8051 microcontroller with suitable examples.**

1. Immediate Addressing: In immediate addressing, the operand is specified directly in the instruction. For example:

MOV A, #25H ; Move immediate value 25H to accumulator A

2. Direct Addressing: In direct addressing, the operand is the contents of a memory location. The address of the memory location is specified in the instruction. For example:

MOV A, 30H ; Move the contents of memory location 30H to accumulator A

3. Register Addressing: In register addressing, the operand is the contents of a register. The register is specified in the instruction. For example:

MOV A, R1 ; Move the contents of register R1 to accumulator A

4. Indirect Addressing: In indirect addressing, the operand is the contents of a memory location whose address is stored in a register. The register is specified in the instruction. For example:

MOV A, @R0 ; Move the contents of the memory location pointed to by R0 to accumulator A

5. Indexed Addressing: Indexed addressing is a variation of indirect addressing where an offset is added to a base register to calculate the address of the operand. For example:

MOV A, @R0+ ; Move the contents of the memory location pointed to by R0 to accumulator A, and increment R0

6. Relative Addressing: Relative addressing is used for branch instructions, where the operand is a signed 8-bit offset that is added to the current program counter to calculate the branch target address. For example:

CJNE A, #10H, Label ; Compare A with immediate value 10H, and branch to Label if not equal.

**5. Explain the TMOD register with respect to timer in 8051.**

The TMOD (Timer Mode) register in the 8051 microcontroller is used to configure the operation mode of the timers. The 8051 has two timers, Timer 0 and Timer 1, and each timer can be configured independently using the TMOD register.

The TMOD register is an 8-bit register located at address 0x89. Its bits are divided into two groups, one for each timer:

- Timer 1 Mode Bits (bits 7-4): These four bits (GATE1, C/T1, M1\_1, M1\_0) are used to configure the operating mode of Timer 1.

- Timer 0 Mode Bits (bits 3-0): These four bits (GATE0, C/T0, M0\_1, M0\_0) are used to configure the operating mode of Timer 0.

Here's a brief explanation of each bit:

1. GATE1 (Timer 1 Gate Control):

- When this bit is set (GATE1 = 1), Timer 1 will only run if the INT1 pin is high (logic 1).

- When this bit is cleared (GATE1 = 0), Timer 1 will run regardless of the state of the INT1 pin.

2. C/T1 (Timer 1 Mode Control):

- When this bit is set (C/T1 = 1), Timer 1 is in timer mode, and it will increment every machine cycle (1/12th of the oscillator frequency).

- When this bit is cleared (C/T1 = 0), Timer 1 is in counter mode, and it will increment on every falling edge of the T1 pin (external input).

3. M1\_1, M1\_0 (Timer 1 Mode Select:

- These two bits are used together to select one of four operating modes for Timer 1:

- Mode 0: 13-bit timer (8-bit auto-reload)

- Mode 1: 16-bit timer

- Mode 2: 8-bit auto-reload timer

- Mode 3: Split timer (two 8-bit timers)

4. GATE0, C/T0, M0\_1, M0\_0 (Timer 0 Mode Control and Select):

- These bits have the same functionality as their Timer 1 counterparts but apply to Timer 0.

Example:

MOV TMOD, #01H ; Timer 1, 16-bit timer, timer mode, no gating

In this example, TMOD is loaded with 01H to configure Timer 1 as a 16-bit timer in timer mode without any gating control.

**6. Differentiate between Microprocessor and Microcontroller.**

1. Architecture:

- Microprocessor: A microprocessor is a single-chip CPU (Central Processing Unit) that only includes the core processing unit. It requires external components such as memory (RAM, ROM), I/O devices, and timers to function as a complete system.

- Microcontroller: A microcontroller is a complete computing system on a single chip. It includes not only the CPU but also memory (RAM, ROM/Flash), I/O ports, timers, and other peripherals necessary for embedded applications.

2. Functionality:

- Microprocessor: Microprocessors are typically used in applications that require high computational power and flexibility. They are commonly found in desktop computers, laptops, and servers.

- Microcontroller: Microcontrollers are designed for embedded applications that require real-time processing and control. They are used in devices such as microcontrollers, industrial control systems, and consumer electronics.

3. System Integration:

- Microprocessor: Microprocessors require additional components such as memory, I/O devices, and peripherals to form a complete system. They are often used in systems where flexibility and scalability are important.

- Microcontroller: Microcontrollers are highly integrated devices that contain all the necessary components on a single chip. They are used in systems where size, cost, and power efficiency are critical factors.

4. Application Examples:

- Microprocessor: Examples of microprocessors include the Intel x86 series used in desktop computers, the ARM processors used in smartphones and tablets, and the MIPS processors used in embedded systems.

- Microcontroller: Examples of microcontrollers include the Atmel AVR series, used in Arduino boards, the Microchip PIC series, and the Texas Instruments MSP430 series, used in various embedded applications.

5. Cost and Power Consumption:

- Microprocessor: Microprocessors tend to be more expensive and consume more power compared to microcontrollers due to their higher complexity and external component requirements.

- Microcontroller: Microcontrollers are typically more cost-effective and have lower power consumption due to their integrated design and targeted application use.

**7. Explain the timer mode 0 and timer mode 1 in 8051 microcontroller.**

In the 8051 microcontroller, Timer 0 and Timer 1 can operate in different modes, which determine how they count and generate interrupts. Timer modes are configured using the TMOD (Timer Mode) register.

Timer Mode 0 (13-bit Timer/Counter Mode):

- In this mode, Timer 0 is configured as a 13-bit timer/counter.

- Timer 0 increments every machine cycle (1/12th of the oscillator frequency).

- The 13-bit timer/counter overflows when it reaches its maximum value (0x1FFF) and then reloads the initial value.

- This mode is often used for generating time delays and baud rate generation.

- Example of configuring Timer 0 in mode 0:

MOV TMOD, #00H ; Timer 0, 13-bit timer, timer mode 0

Timer Mode 1 (16-bit Timer/Counter Mode):

- In this mode, Timer 1 is configured as a 16-bit timer/counter.

- Timer 1 increments every machine cycle (1/12th of the oscillator frequency).

- The 16-bit timer/counter overflows when it reaches its maximum value (0xFFFF) and then reloads the initial value.

- This mode is often used for generating time delays and measuring time intervals.

- Example of configuring Timer 1 in mode 1:

MOV TMOD, #20H ; Timer 1, 16-bit timer, timer mode 1

In both modes, the timer/counters can be used in conjunction with interrupts to generate periodic interrupts based on the timer's overflow. The timer modes and their configurations allow for flexible use of the timers in various timing and control applications in 8051-based systems.

**8. Explain situations when microprocessor based operations are preferred over microcontroller based operations**.

Microprocessor-based operations are preferred over microcontroller-based operations in situations where:

- Flexibility is required: Microprocessors are more flexible and can be used in a wide range of applications due to their ability to interface with external memory and peripherals. This flexibility allows for the implementation of complex algorithms and protocols.

- High computational power is needed: Microprocessors are designed for applications that require high computational power, such as desktop computers, servers, and high-performance computing systems. They typically have faster clock speeds and larger instruction sets compared to microcontrollers.

- Customization is necessary: Microprocessors allow for more customization in terms of hardware and software. Developers can choose the specific components and peripherals they need for their application, making them ideal for specialized or niche applications.

- Cost is not a primary concern: Microprocessors tend to be more expensive than microcontrollers due to their higher complexity and external component requirements. In situations where cost is not a primary concern, microprocessors may be preferred for their performance and flexibility.