ECT 206 COMPUTER ARCHITECTURE AND MICROCONTROLLERS*

SOUMYA S
AP IN ECE
CEMP

8051 – Interrupts

- An interrupt is an internal or external event that interrupt the microcontroller to inform it that a device needs its service
- A microcontroller can serve devices in two ways

—>> Interrupts

—>> Polling

8051 – Interrupts

- Interrupts
  - Whenever a device needs its service, the device notifies the microcontroller by sending it an interrupt signal
  - On receiving the interrupt signal, the microcontroller interrupts whatever it is doing and serve the device
  - The program which is associated with the interrupt is called Interrupt Service Routine (ISR) or Interrupt Handler

Module 2-8051 Architecture cont'

8051 – Interrupts

- Polling
  - The microcontroller continuously monitors the status of a given device
  - When a condition met, it performs the service
  - After that, it moves on to monitor the next device until everyone is serviced
  - Polling is not efficient, because it waste most of the processor time, by polling device that do not need service

Module 2-8051 Architecture cont'

051 – Interrupts -Advantage

- Microcontroller can serve many devices
- Each device can get attention of the microcontroller based on the priority assigned.
- For the polling method it is not possible to assign priority because it check all device one by one.
- Microcontroller can ignore a device interrupt request(Masking)

8051 Interrupts

- An interrupt is an event that occurs randomly in the flow of continuity. It is just like a call you have when you are busy with some work and depending upon call priority you decide whether to attend or neglect it.
- The same thing happens in microcontrollers. 8051 architecture handles 5 interrupt sources, out of which two are internal (Timer Interrupts), two are external and one is a serial interrupt. Each of these interrupts has its interrupt vector address. The highest priority interrupt is the Reset, with vector address 0x0000.

- **Vector Address**: This is the address where the controller jumps after the interrupt to serve the ISR (interrupt service routine).

Interrupt	Flag	Interrupt vector address
Reset	-	0000H
INT0 (Ext. int. 0)	IE0	0003H
Timer 0	TF0	000BH
INT1 (Ext. int. 1)	IE1	0013H
Timer 1	TF1	001BH
Serial	TI/RI	0023H

- **RESET interrupt** – This is also known as Power on Reset (POR). When the RESET interrupt is received, the controller restarts executing code from 0000H location. This is an interrupt which is not available to or, better to say, need not be available to the programmer.

- **2. Timer interrupts** – Each Timer is associated with a Timer interrupt. A timer interrupt notifies the microcontroller that the corresponding Timer has finished counting.

- **3. External interrupts** – There are two external interrupts EX0 and EX1 to serve external devices. Both these interrupts are active low. In AT89C51, P3.2 (INT0) and P3.3 (INT1) pins are available for external interrupts 0 and 1 respectively. An external interrupt notifies the microcontroller that an external device needs its service.

- **4. Serial interrupt** – This interrupt is used for serial communication. When enabled, it notifies the controller whether a byte has been received or transmitted.

Steps in executing an interrupt

Upon activation of an interrupt, the microcontroller goes through the following steps.

1. It finishes the instruction it is executing and saves the address of the next instruction (PC) on the stack.
2. It also saves the current status of all the registers.
3. It jumps to a fixed location in memory called the interrupt vector table that holds the address of the interrupt service routine.
4. The microcontroller gets the address of the ISR from the interrupt vector table and jumps to it. It starts to execute the interrupt service subroutine until it reaches the last instruction of the subroutine, which is RETI (return from inter rupt).
5. Upon executing the RETI instruction, the microcontroller returns to the place where it was interrupted. First, it gets the program counter (PC) address from the stack by popping the top two bytes of the stack into the PC. Then it starts to execute from that address.

- 8051 makes use of 2 registers to deal with interrupts.
- IE – Interrupt Enable.
- IP – Interrupt Priority.

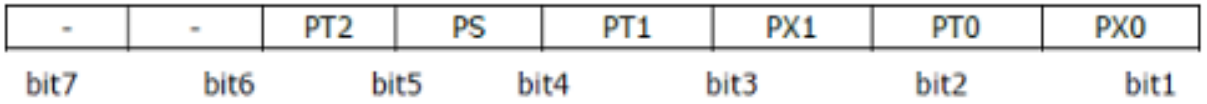
- This register is responsible for enabling and disabling the interrupt. EA register is set to one for enabling interrupts and set to 0 for disabling the interrupts. Its bit sequence and their meanings are shown in the following figure.

EA	-	-	ES	ET1	EX1	ET0	EX0

EA	IE.7	It disables all interrupts. When EA = 0 no interrupt will be acknowledged and EA = 1 enables the interrupt individually.
-	IE.6	Reserved for future use.
-	IE.5	Reserved for future use.
ES	IE.4	Enables/disables serial port interrupt.
ET1	IE.3	Enables/disables timer1 overflow interrupt.
EX1	IE.2	Enables/disables external interrupt1.
ET0	IE.1	Enables/disables timer0 overflow interrupt.
EX0	IE.0	Enables/disables external interrupt0.

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.



-	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.