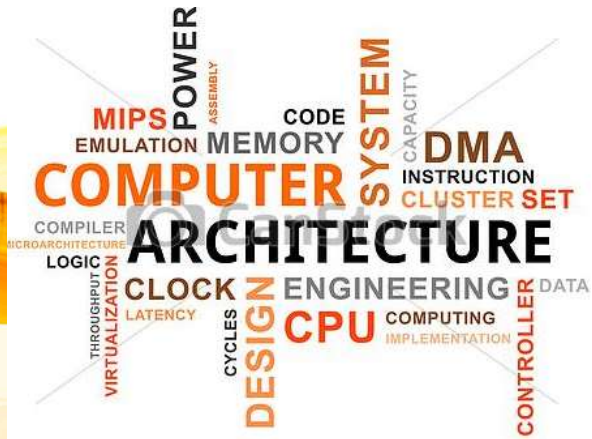
UNIT V
I/O ORGANIZATION AND PARALLELISM

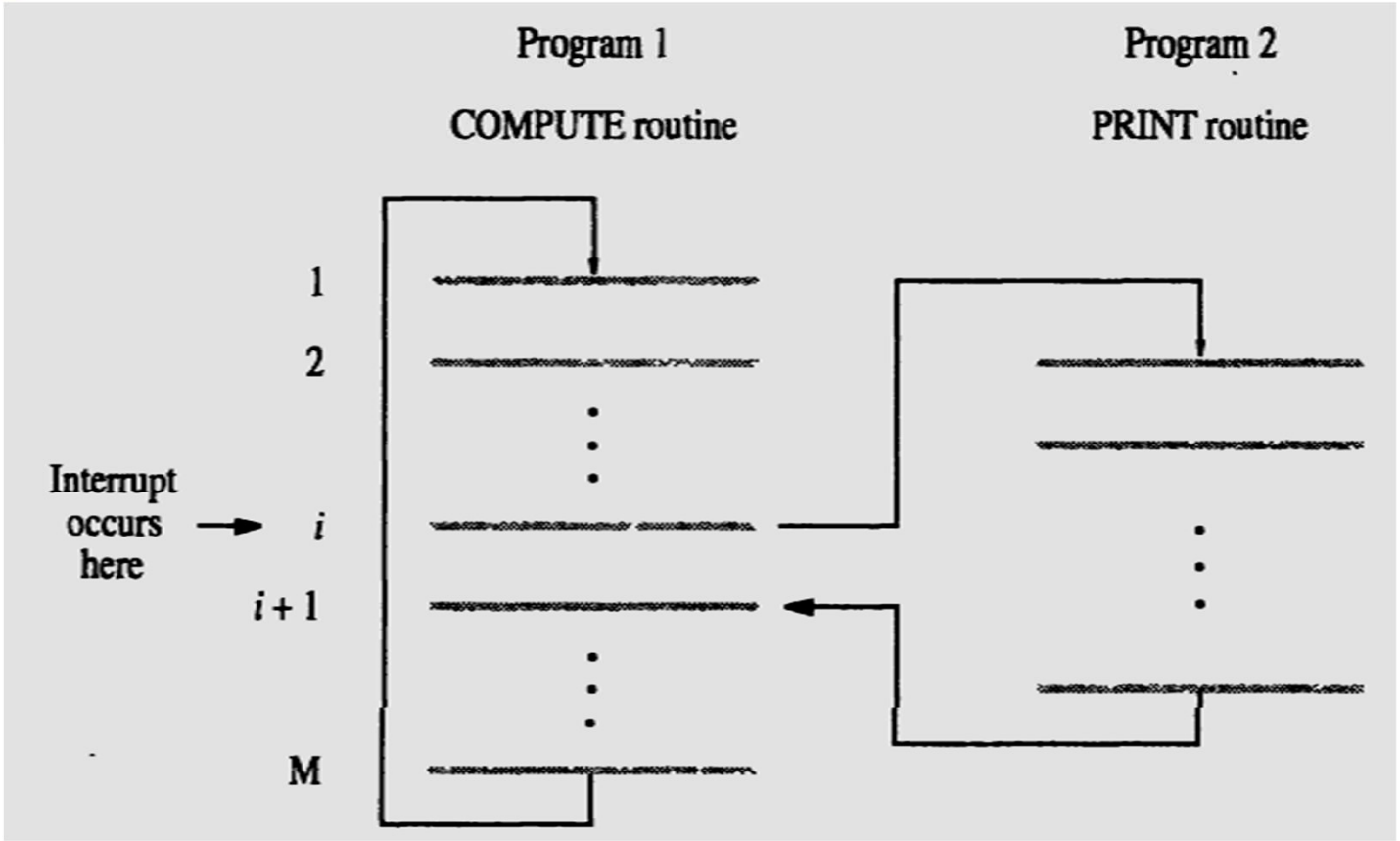
Accessing I/O devices – **Interrupts** – Direct Memory Access - Buses– Interface circuits - Standard I/O Interfaces (PCI, SCSI, USB)–Instruction Level Parallelism : Concepts and Challenges – Introduction to multicore processor Graphics Processing Unit.



Recap the previous Class

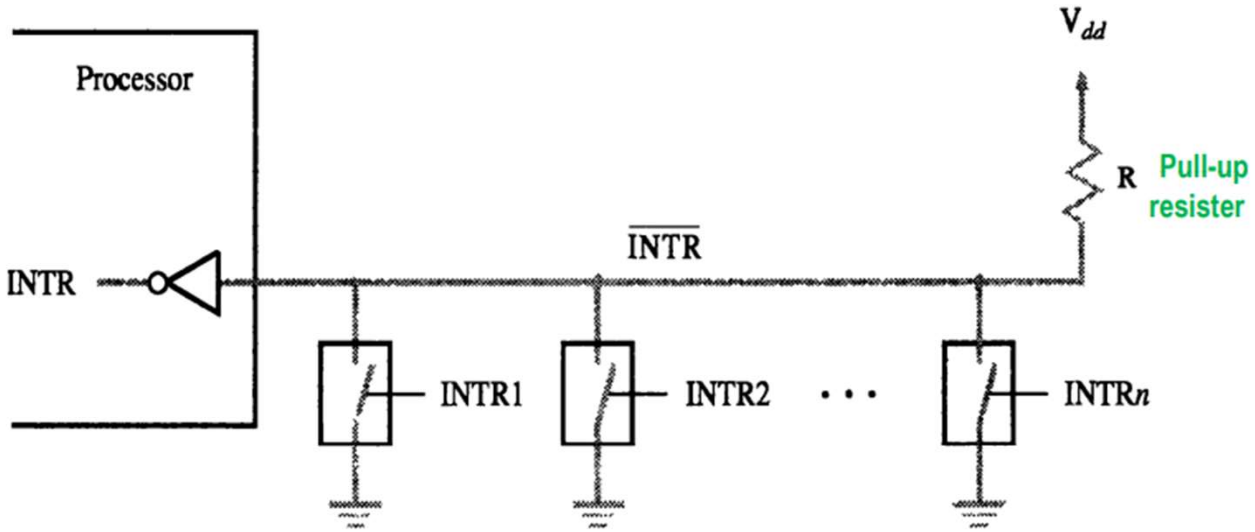
Interrupts

- When I/O Device is ready, it sends the INTERRUPT signal to processor via a dedicated controller line
- Using interrupt, we are ideally eliminating WAIT period
- In response to the interrupt, the processor executes the Interrupt Service Routine (ISR)
- All the registers, flags, program counter values are saved by the processor before running ISR
- The time required to save status & restore contribute to execution overhead "Interrupt Latency"


Program 1
COMPUTE routine
Program 2
PRINT routine
1
2
Interrupt occurs here
i
i + 1
M

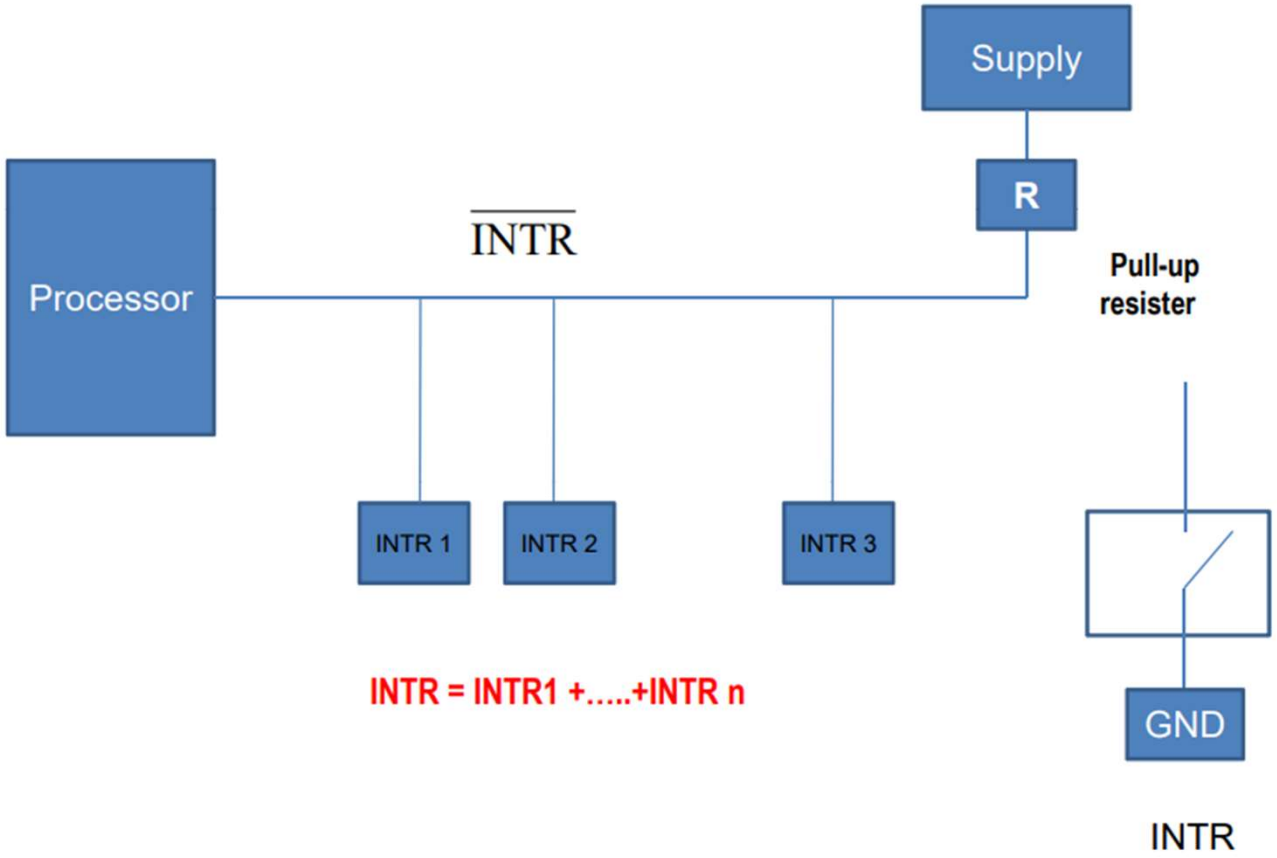
- interrupt-acknowledge signal - I/O device interface accomplishes this by execution of an instruction in the interrupt-service routine (ISR) that accesses a status or data register in the device interface; implicitly informs the device that its interrupt request has been recognized. IRQ signal is then removed by device.
- ISR is a sub-routine – may belong to a different user than the one being executed and then halted.
- The condition code flags and the contents of any registers used by both the interrupted program and the interrupt-service routine are saved and restored.
- The concept of interrupts is used in operating systems and in many control applications, where processing of certain routines must be accurately timed relative to external events (e.g. real-time processing).

Interrupt Hardware



An equivalent circuit for an open-drain bus used to implement a common interrupt-request line

Interrupt Hardware



Enabling and Disabling Interrupts

- Device activates interrupt signal line and waits with this signal activated until processors attends
- The interrupt signal line is active during execution of ISR and till the device caused interrupt is serviced
- Necessary to ensure that the active signal does not lead to successive interruptions (level-triggered input) causing the system to fall in infinite loop.
- What if the same device interrupts again, within an ISR ?
- Three methods of Controlling Interrupts (single device)
  - Ignoring interrupt
  - Disabling interrupts
  - Special Interrupt request line

- **Ignoring Interrupts**
  - Processor hardware ignores the interrupt request line until the execution of the first instruction of the ISR completed
  - Using an interrupt disable instruction after the first instruction of the ISR – no further interrupts
  - A return from interrupt instruction is completed before further interruptions can occur

- **Disabling Interrupts**
  - Processor automatically disables interrupts before starting the execution of the ISR
  - The processor saves the contents of PC and PS (status register) before performing interrupt disabling.
  - The interrupt-enable is set to 0 – no further interrupts allowed
  - When return from interrupt instruction is executed the contents of the PS are restored from the stack, and the interrupt enable is set to 1

- **Special Interrupt line**
  - Special interrupt request line for which the interrupt handling circuit responds only to the leading edge of the signal
  - Edge –triggered
  - Processor receives only one request regardless of how long the line is activated
  - No separate interrupt disabling instructions

The sequence of events involved in handling an interrupt request from a single device.

Assuming that interrupts are enabled, the following is a typical scenario:

1. The device raises an interrupt request.

2.The processor interrupts the program currently being executed.

3.Interrupts are disabled by changing the control bits in the PS (except in the case of edge-triggered interrupts).

4.The device is informed that its request has been recognized, and in response, it deactivates the interrupt- request signal.

5.The action requested by the interrupt is performed by the interrupt-service routine.

6.Interrupts are enabled and execution of the interrupted program is resumed.

Handling Multiple Devices

- Multiple devices can initiate interrupts
- They uses the common interrupt request line
- Techniques are
  - Polling
  - Vectored Interrupts
  - Interrupt Nesting
  - Daisy Chaining

Polling Scheme

- The IRQ (interrupt request) bit in the status register is set when a device is requesting an interrupt.
- The Interrupt service routine polls the I/O devices connected to the bus.
- The first device encountered with the IRQ bit set is serviced and the subroutine is invoked.
- Easy to implement, but too much time spent on checking the IRQ bits of all devices, though some devices may not be requesting service

Vectored Interrupts

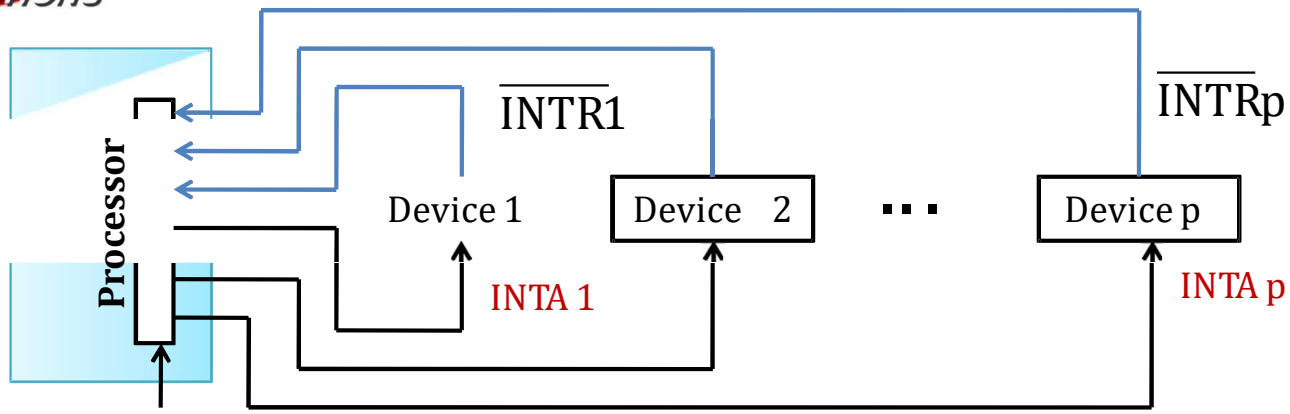
- Device requesting an interrupt identifies itself directly to the processor
- The device sends a special code to the processor over the bus.
- The code contains the
  - identification of the device,
  - starting address for the ISR,
  - address of the branch to the ISR
- PC finds the ISR address from the code.
- To add flexibility for multiple devices - corresponding ISR is executed by the processor using a branch address to the appropriate routine - device specified *Interrupt Vector*.

- An interrupt vector is the memory address of an interrupt handler, or an index into an array called an interrupt vector table or dispatch table - a table of interrupt vectors (pointers to routines that handle interrupts).
- Interrupt vector tables contain the memory addresses of interrupt handlers. When an interrupt is generated, the processor saves its execution state via a context switch, and begins execution of the interrupt handler at the interrupt vector.
- The Interrupt Descriptor Table (IDT) is specific to the I386 architecture. It tells where the Interrupt Service Routines (ISR) are located.
- Each interrupt number is reserved for a specific purpose. For example, 16 of the vectors are reserved for the 16 IRQ lines.
- On PCs, the interrupt vector table (IVT or IDT) consists of 256 4-byte pointers - the first 32 (0-31 or 00-1F) of which are reserved for processor exceptions; the rest for hardware interrupts, software interrupts. This resides in the first 1 K of addressable memory.

Interrupt Nesting

- Pre-Emption of low priority Interrupt by another high priority interrupt is known as Interrupt nesting.
- Disabling Interrupts during the execution of the ISR may not favor devices which need immediate attention.
- Need a priority of IRQ devices and accepting IRQ from a high priority device.
- The priority level of the processor can be changed dynamically.
- The privileged instruction write in the PS (processor status word), that encodes the processors priority.

Interrupt Nesting (contd.)

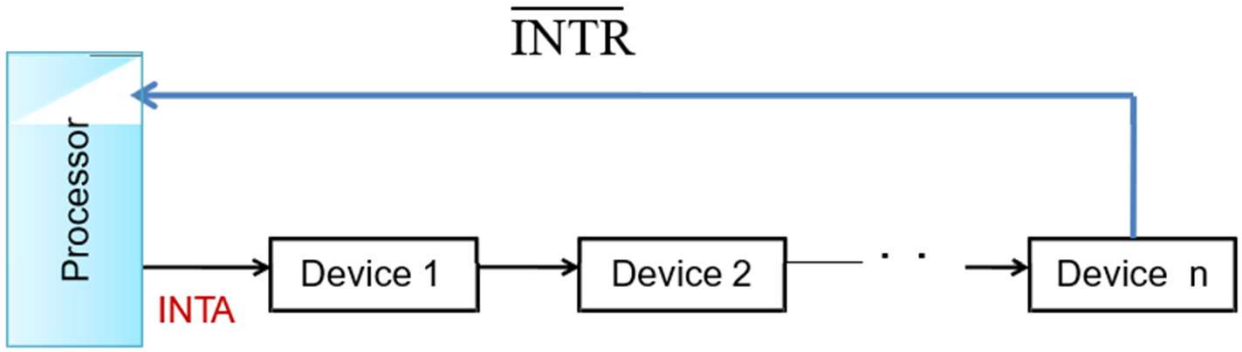


Priority arbitration circuit

- Organizing I/O devices in a prioritized structure.
- Each of the interrupt-request lines is assigned a different priority level.
- The processor is interrupted only by a high priority device.

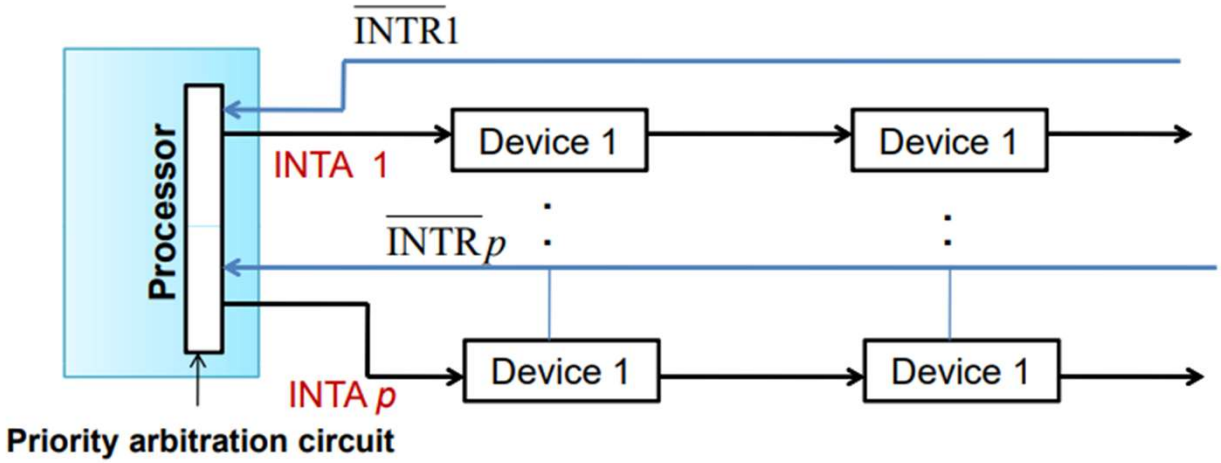
Daisy Chaining

- The interrupt request line INTR is common to all the devices
- The interrupt acknowledgement line INTA is connected to devices in a DAISY CHAIN way
- INTA propagates serially through the devices
- Device that is electrically closest to the processor gets high priority
- Low priority device may have a danger of STARVATION



Daisy Chaining with Priority Group

- Combining Daisy chaining and Interrupt nesting to form priority group
- Each group has different priority levels and within each group devices are connected in daisy chain way



Arrangement of priority groups


Thank You