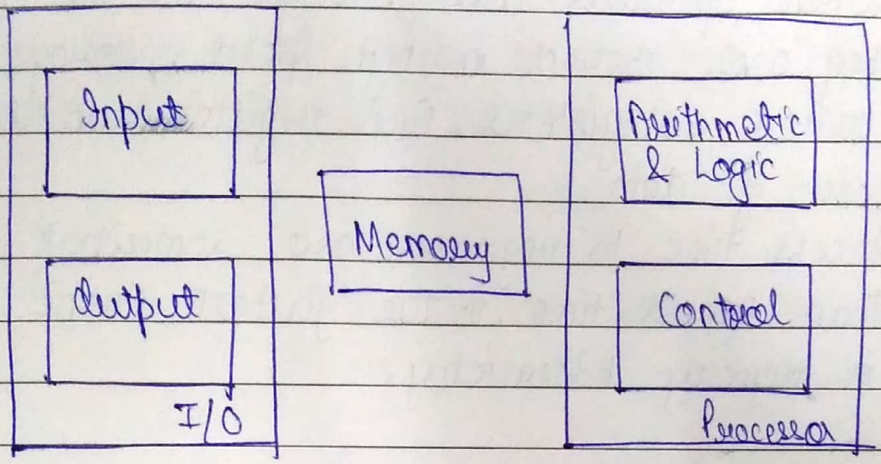
UNIT-1

JAY PRAKASH Date_____ Page_____

Functionally independent main parts:

- input
- memory
- arithmetic and logic units
- output
- control unit



Memory Unit

Primary Memory

It is a fast memory that operates at electronic speeds. It contains a large number of semiconductor storage cells, each capable of storing one bit of information. These cells are rarely read or written as individual cells but instead are processed in groups of fixed sizes called words. The memory is organized so that the contents of one word, containing n bits, can be stored or retrieved in one basic operation.

A distinct address is associated with each word location.

JAY PRAKASH Date Page

Arithmetic and Logic Unit

Any arithmetic or logic operation is initiated by bringing the required operands into the processor, where the operation is performed by the ALU.
When operands are brought into the processor, they are stored in high speed storage elements called registers. Each register can store one word of data.
Access time to registers are somewhat faster than access time to the fastest cache unit in memory hierarchy.

Control Unit

The operations of all the functional units are controlled by this unit.
Control unit is a well defined, physically separate unit that interacts with other parts of the machine.
A large set of control lines (wires) carries the signals used for timing and synchronization of events in all units.

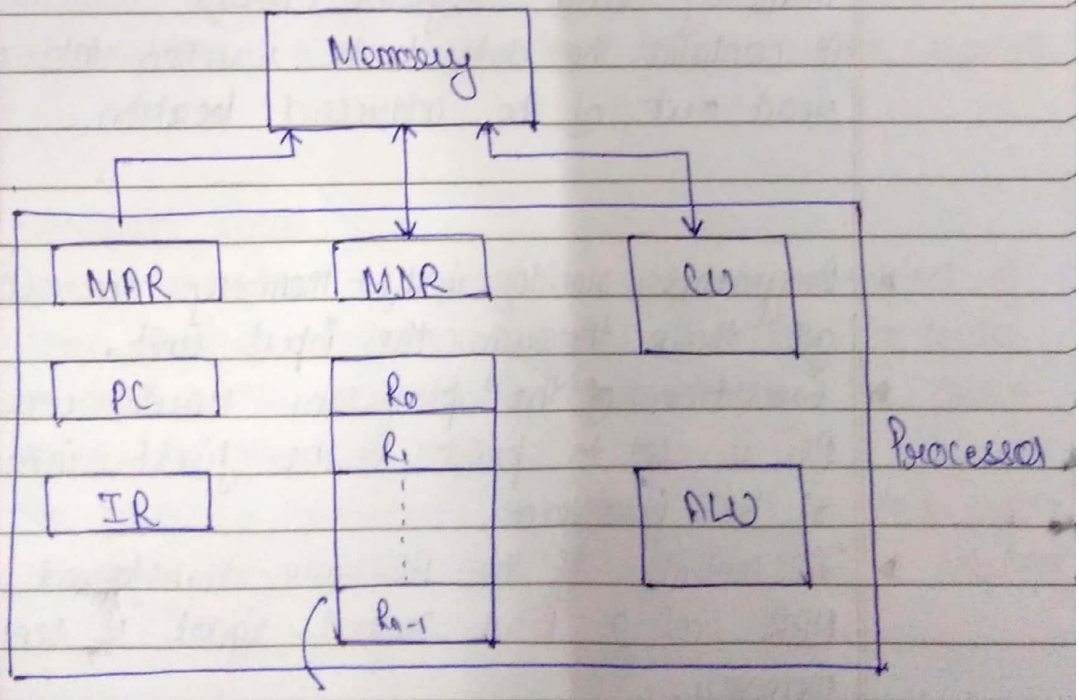
JAY PRAKASH

Basic Operational Concepts

Add Loca, R_0 → the instruction adds the operand at memory location 'Loca' to the operand in the processor, R_0, and places the sum into the register R_0.

The original contents of location Loca are preserved, whereas those of R_0 are overwritten.

Load Loca, R_1 } → this destroys the original
Add R_1, R_0 } contents registeres R_0 as well as R_1.
The original contents of Loca are preserved.



- Instruction Register (IR)
  It holds the instruction that is currently being executed. Its output is available to the control circuits, which generate the timing signals that controls the various processing elements.

- Program Counter (PC)
  It keeps track of the execution of the program. It contains the memory address of the next instruction to be fetched and executed.

- Memory Address Register (MAR)
  It holds the address of the location to be accessed.

- Memory Data Register (MDR)
  It contains the data to be written into or read out of the addressed location.

- Programs reside in the memory and usually get there through the input unit.
- Execution of the program starts when the PC is set to point to the first instruction of the program.
- The contents of the PC are transferred to the MAR and a Read control signal is sent to memory.
- After the time required to access the memory

elapses, the addressed word (in this case, the first instruction of the program) is read out of the memory and loaded into the MBR.

- Next, the contents of the MBR are transferred to the IR. At this point, the instruction is ready to be decoded and executed.
- If the instruction involves an operation to be performed by the ALU, it is necessary to obtain the required operands. If an operand resides in the memory, it has to be fetched by sending its address to the MAR and initiating a Read cycle. When the operand has been read from the memory into the MBR, it is transferred to the ALU. If the result is to be stored in the memory, then the result is sent to the MBR. The address of the location where the result is to be stored is sent to MAR.
- As soon as the execution of the current instruction is completed, a new instruction fetch may be started.

Interrupt Signal

Normal execution of program may be preempted if some device requires urgent servicing. In order to deal with the situation immediately, the device raises an interrupt signal.

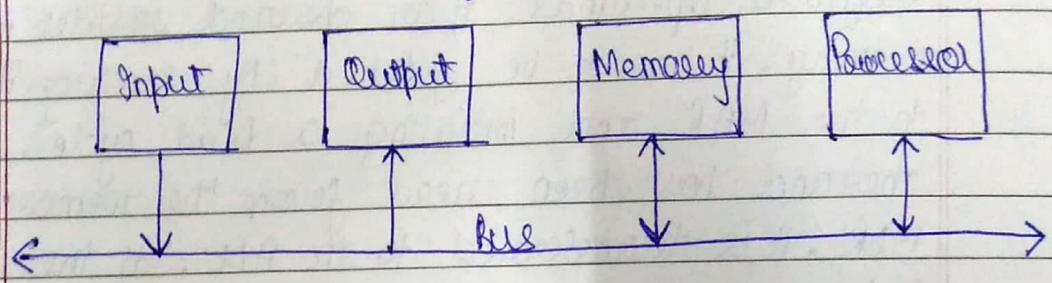
The processor provides the requested service by executing an appropriate interrupt-service routine.

JAY PRAKASH Date Page

Bus Structures

A group of lines that serves as a connecting path for several devices is called a bus.

The bus can be used for only one transfer at a time, hence only two units can actively use the bus at any given time.



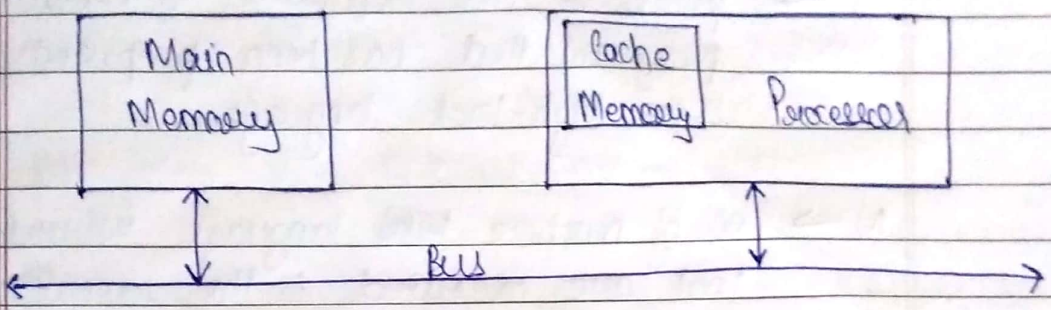
The main virtue of the single-bus structure is its low cost and its flexibility for attaching peripheral devices.

Buffer Registers

They are used with devices to hold the information during transfers.
Buffer registers smooth out timing differences among processors, memories and I/O devices.
It allows the processor to switch rapidly from one device to another.

JAY PRAKASH Date Page

Performance



The Processor Cache

A program will be executed faster if the movement of instructions and data between the main memory and the processor is minimized, which is achieved by using the cache.

Processor clock

Processor circuits are controlled by a timing signal called a clock.

The clock defines regular time intervals, called clock cycles.

To execute a machine instruction, the processor divides the action to be performed into a sequence of basic steps, such that each step can be completed in one clock cycle.

The length P of one clock cycle is an important parameter that affects processor performance. Its inverse is the clock rate R.

$$R = \frac{1}{P} \text{ Hz}$$

JAY PRAKASH Date Page

Basic Performance Equation

T → processor time required to execute a program that has been prepared in some high-level language.

N → no. of machine language instructions that are executed in the execution of one complete program.

S → average number of basic steps needed to execute one machine instruction, where each basic step is completed in one clock cycle.

R → clock rate (in cycles per second)

$$\therefore T = \frac{N \times S}{R}$$

To achieve higher performance, T must be decreased. This is achieved by reducing N, and S and increasing R.

The value of N is reduced if the source code is compiled into fewer machine instructions.

The value of S is reduced if instructions have a smaller no. of basic steps performed or if the execution of instruction is overlapped.

The value of R is increased by using higher frequency clock which reduces the execution time of a basic step.

Pipelining and Superscalar

Pipelining is the overlapping of execution of successive instructions.

Add R_1, R_2, R_3

The above command adds the contents of registers R_1 and R_2 and places the sum into R_3.

The contents of R_1 and R_2 are first transferred to the inputs of ALU. After the add operation is performed, the sum is transferred to R_3. The processor can read the next instruction from the memory while the addition operation is being performed.

A higher degree of concurrency is achieved if multiple instruction pipelines are implemented in the processor. This means that multiple functional units are used, creating parallel paths through which different instructions can be executed in parallel. Thus several instructions are executed in every clock cycle. This is called as Superscalar Execution.

Clock Rate

There are two ways to increase the clock rate (R):

- Improving the IC technology, making logic circuit faster.
- Reducing the amount of processing done in one basic step.

RISC – Reduced Instruction Set Computers
CISC – Complex Instruction Set Computers

Performance Measurement

The performance of a computer is measured using benchmark programs.
The time taken to execute a benchmark is the performance measure.

SPEC → System Performance Evaluation Corporation

$$\text{SPEC Rating} = \frac{\text{Running time on reference computer}}{\text{Running time on computer under test}}$$

A SPEC rating of 50 means that the computer under test is 50 times as fast as the reference computer.

SPEC rating is a measure of the combined effect of all factors affecting performance.

Compiler
It translates a high level language program into a sequence of machine instructions.
An optimizing compiler takes advantage of the various features of the target processor to reduce the product N x S, which is the total no. of clock cycles needed to execute a program.
The compiler may rearrange program instructions to achieve better performance.

Multiprocessors & Multicomputers

Large computer systems containing a large no. of processor units are called multiprocessor systems.
These systems either execute a number of different application tasks in parallel or they execute subtasks of a single large task in parallel.

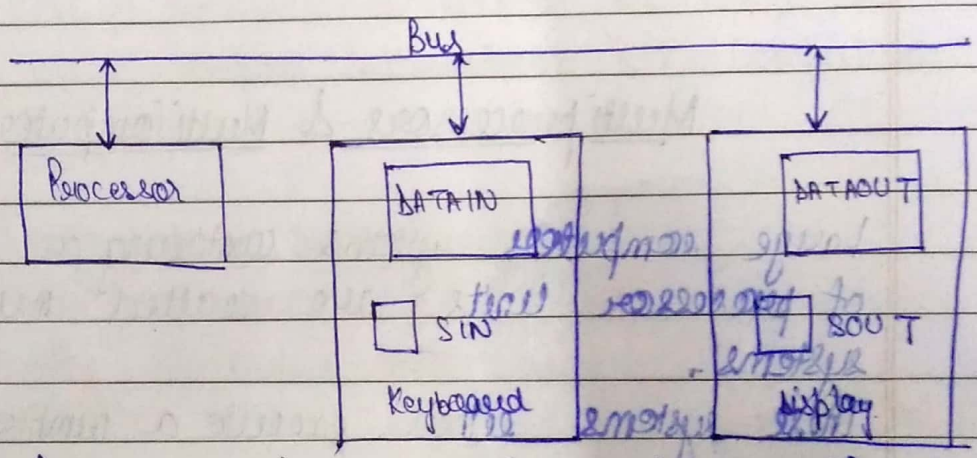
Shared Memory Multiprocessor Systems -
It means that all the processors in a multiprocessor system have access to all of the memory in the system.

Multicomputer system is a group of interconnected complete computers.
When the task they are executing need to communicate data, they do so by exchanging

messages over a communication network.

Message Passing Multicomputer Systems -
The computers normally has access only to their own memory units, thus there is a need to exchange messages between computers.

Basic Input/Output Operations



Bus connection for processor, keyboard & display

A simple way of performing basic I/O tasks is to use a method known as program-controlled I/O.
The rate of data transfer from the keyboard to a computer is limited by the typing speed of the user, which is unlikely to exceed a few characters per second.
The rate of output transfers from the computer to the display is much higher.
It is still slower than the speed of a processor that can execute millions of

instructions per second. The difference in speeds between the processor & I/O devices creates the need for mechanisms to synchronise the transfer of data between them.

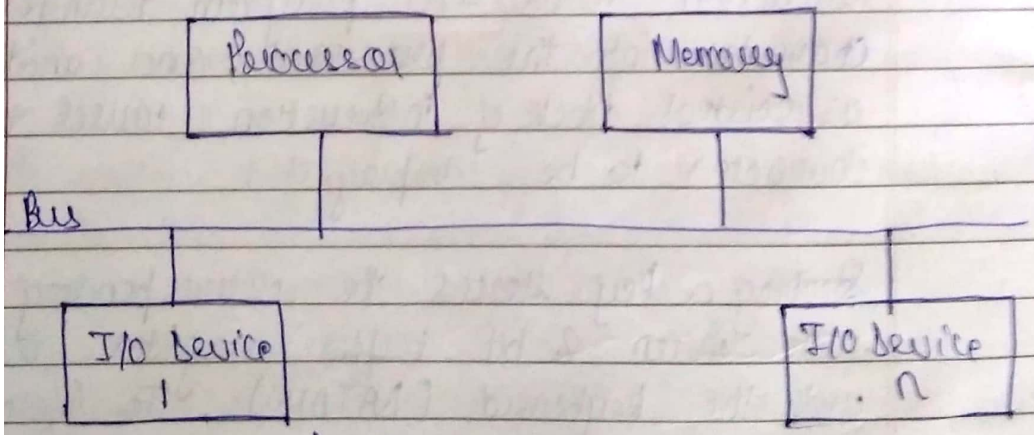
A solution to this problem is as follows: On output, the processor sends the first character and then waits for the signal from the display that the character has been received. It then sends the character and so on. Input from the keyboard is sent in a similar way. The processor waits for a signal indicating that a character has been struck and that its code is available in some buffer register.
The action of striking a key does not automatically cause the corresponding character to be displayed. One block of instruction in the I/O program transfers the character into the processor, and another associated block of instruction causes the character to be displayed.

Striking a key stores the corresponding character code in an 8-bit buffer register associated with the keyboard (DATAIN). To inform the processor that a valid character is in DATAIN, a status control flag, SIN is set to 1. A program monitors SIN, and when SIN is set to 1, the processor reads the character from DATAIN. When the character is transferred to the processor, SIN is automatically cleared to 0.

An analogous process takes place when characters are transferred from the processor to the display. When the status control flag SOUT equals to 1, the character is transferred to the buffer register DATAOUT. The transfer of data to DATAOUT clears SOUT to 0. When the display device is ready to receive another character, SOUT is again set to 1.

The buffer registers DATAIN & DATAOUT and the status flags SIN & SOUT are part of circuitry known as device interface.

Accessing I/O Devices

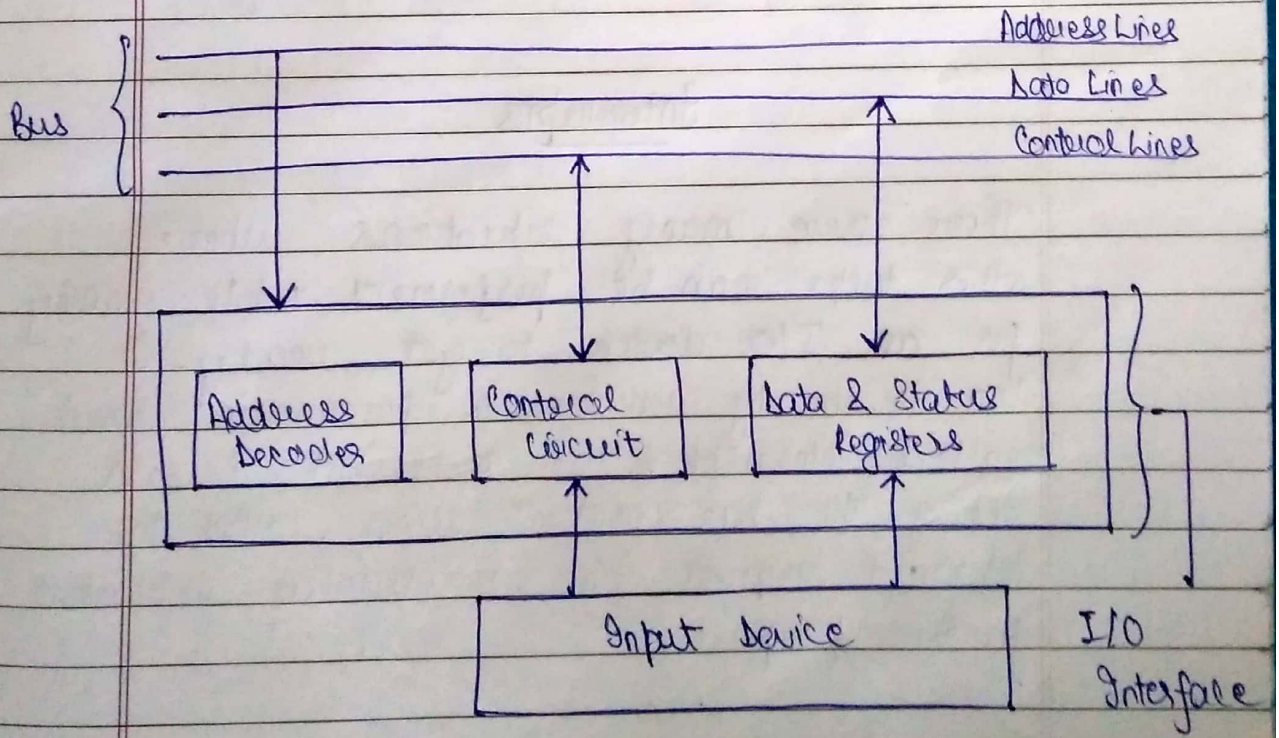


A single bus structure

A bus enables all the devices connected to it to exchange information.
Typically, it consists of three sets of lines used to carry address, data and control

signals. Each I/O device is assigned a unique set of address. When the processor places a particular address on the address line, the device that recognizes this address responds to the commands issued on the control lines. The processor requests either a read or write operation, and the requested data is transferred over the data lines.

Memory Mapped I/O : When I/O devices and the memory share the same address space. In this, any machine instruction that can access memory can be used to transfer data to or from an I/O device.



I/O Interface for an Input Device

The address decoder enables the device to recognise its address when this address appears on the address line. The data registers holds the data being transferred to or from the processor. The status registers contains the information relevant to the operation of the I/O device. Both the data & status registers are connected to the data bus and are assigned unique addresses.

The address decoder, data & status registers and the control circuitry required to coordinate I/O transfers constitute the device's interface circuit.

Interrupts

There are many situations where other tasks can be performed while waiting for an I/O device to get ready. It can do so by sending a hardware signal called interrupt to the processor. Atleast one of the bus control lines, called the interrupt request line, is usually dedicated for this purpose.

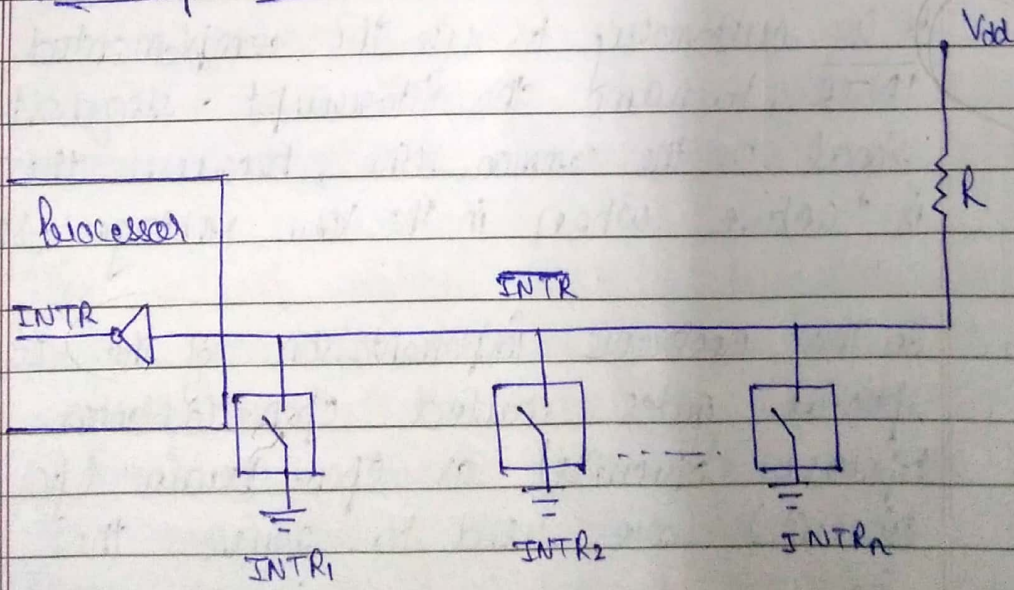
The routine executed in response to an interrupt request is called interrupt-service routine.

As part of handling interrupts, the processor must inform the device that its request has been recognised so that it may remove its interrupt-service request signal. This may be achieved by a special control signal on the bus, called interrupt-acknowledge signal.

A subroutine performs a function required by the program from which it is called.
Interrupt service routine may not have anything in common with the program being executed.

Interrupt Latency - The process of saving and restoring registers involve memory transfers that increase the execution time. This delay is called interrupt latency.

Interrupt Hardware



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

A single interrupt-request line may be used to serve n devices.
All devices are connected to the line via switches to ground. To request an interrupt, a device closes its associated switch. Thus if all interrupt-request signals INTR1 to INTRn are inactive i.e. if all switches are open, the voltage on the request line will be equal to Vdd.
When a device requests an interrupt by closing its switch, the voltage of the line drops to 0, causing the interrupt request signal, INTR, received by the processor to go to 1. Since the closing of one or more switches will cause the line voltage to drop to 0, the value of INTR is the logical OR of the requests from individual devices, i.e.:

$$INTR = INTR_1 + \cdots + INTR_n$$

It is customary to use the complemented form, $\overline{INTR}$, to name the interrupt-request signal on the common line, because this signal is active when in the low voltage state.

In the electronic implementation of the circuit, special gates called open-collector (for bipolar circuits) or open-drain (for MOS circuits) are used to drive the $\overline{INTR}$ line.
Resistor R is called a pull-up resistor because it pulls the line voltage up to the

high voltage state when the switches are open.

Enabling and Disabling Interrupts

The first possibility is to have the processor hardware ignore the interrupt request line until the execution of the first instruction of the interrupt service routine has been completed. Then by using an interrupt disable instruction as the first instruction in the interrupt-service routine, that no further interruptions will occur until an interrupt-enable instruction is executed. Typically, the interrupt-enable instruction will be the last instruction in the interrupt-service routine before the return-from-interrupt instruction. The processor must guarantee that execution of the Return-from-interrupt is completed before further interruption can occur.

The second option, which is suitable for a simple processor with only one interrupt-request line, is to have the processor automatically disable interrupts before starting the execution of the interrupt-service routine. After saving the contents of the PC and the processor status register (PS) on the stack, the processor performs the equivalent of

executing an interrupt-service routine. It is often the case that one bit in the PS register, called interrupt-enable, indicates whether interrupts are enabled. An interrupt request received while this bit is equal to 1 will be accepted. After saving the contents of the PS on the stack, with the interrupt-enable bit equal to 1, the processor clears the interrupt enable bit in its PS register, thus disabling further interrupts. When a return-from-interrupt instruction is executed, the contents of the PS are restored from the stack, setting the interrupt-enable bit back to 1.

In the third option, the processor has a special interrupt-request line for which the interrupt handling circuit responds only to the leading edge of the signal. Such a line is said to be edge triggered. In this case, the processor will receive only one request, regardless of how long the line is activated.

* Sequence of events involved in handling an interrupt (assuming interrupts are enabled):
  - The device raises an interrupt request.

- The processor interrupts the program currently being executed.
- Interrupts are disabled by changing the control bits in the PS (except in edge triggered interrupts)
- The device is informed that its request has been recognized, and in response, it deactivates the interrupt-request signal.
- The action requested by the interrupt is performed by the interrupt-service routine.
- Interrupts are enabled and the execution of the interrupted program is resumed.

Handling Multiple Devices

The information needed to determine whether a device is requesting an interrupt is available in its status register. When a device raises an interrupt request, it sets to 1, one of the bits in its status register, called the IRQ bit.

The simplest way to identify the interrupting device is to have the interrupt-service routine poll all the I/O devices connected to the bus. The first device encountered with its IRQ bit set is the device that should be serviced. An appropriate subroutine is called to provide the requested service.

The polling scheme is easy to implement. Its main disadvantage is the time spent interrogating the I/O bits of all the devices that may not be requesting any service.

* Vectored Interrupts :

To reduce the time involved in the polling process, a device requesting an interrupt may identify itself directly to the processor.

A device requesting an interrupt can identify itself by sending a special code to the processor over the bus. This enables the processor to identify individual devices even if they share a single interrupt-request line. The code supplied by the device may represent the starting address of the interrupt-service routine for that device. The code length is in the range of 4 to 6 bits.

This arrangement implies that the interrupt-service routine for a given device must start at the same location. The location pointed to by the interrupting device is used to store the starting address of the interrupt-service routine. The processor reads this address, called the interrupt vector, and loads it into the PC.

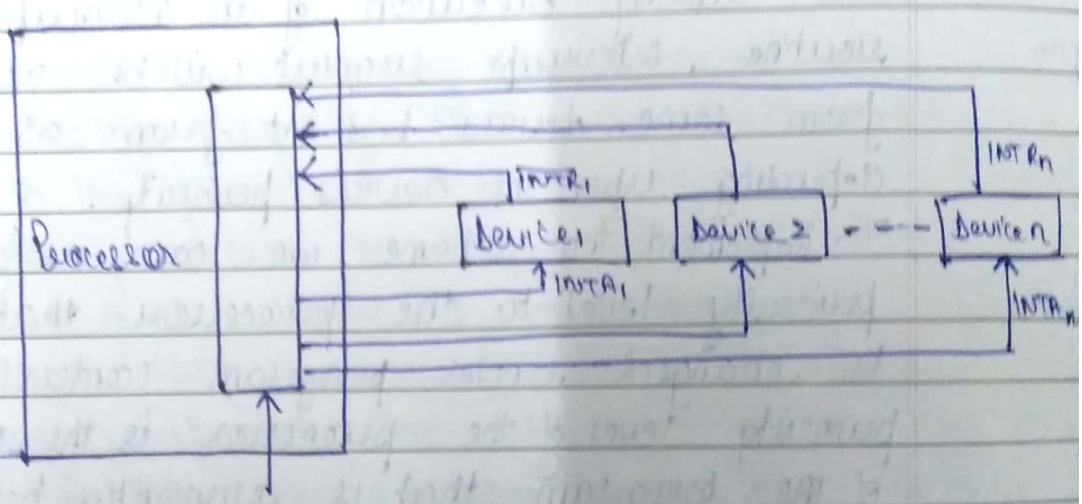
* Interrupt Nesting:

When several devices are involved, in which case execution of a given interrupt-service routine, once started, always continues to completion before the processor accepts an interrupt request from a second device. In some cases, a long delay in corresponding to an interrupt request may lead to erroneous operation.

A multiple-level priority organization means that during execution of an interrupt service routine, interrupt request will be accepted from some devices but not from others, depending upon the device's priority.
To implement this scheme, we can assign a priority level to the processor that can be changed under program control. The priority level of the processor is the priority of the program that is currently being executed. The processor accepts interrupts only from devices that have priority higher than its own. At the time the execution of an interrupt-service routine for some device is started, the priority of the processor is raised to that of the device.

The processor's priority is usually encoded in a few bits of the processor status word. It can be changed by program instructions that write into the PS. These are privileged instructions

which can be executed only while the processor is running in supervisor mode. The processor is in the supervisor mode only when running operating system routines and switches to user mode to execute application programs.
An attempt to execute a privileged instruction while in the user mode leads to a special type of interrupt called a privilage exception.



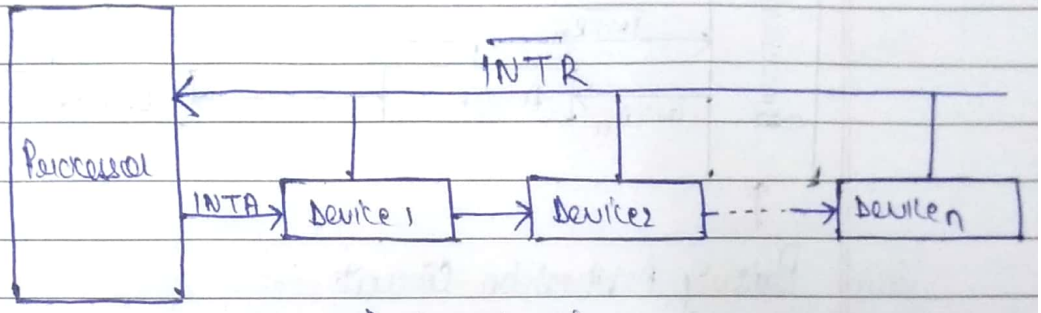
Priority Arbitration Circuit

fig- Implementation of Interrupts priority using individual interrupt request and acknowledge lines

JAY PRAKASH Date Page

* Simultaneous Requests

To solve the problem of simultaneous arrivals of interrupt requests from two or more devices, the devices are connected to form a daisy chain.

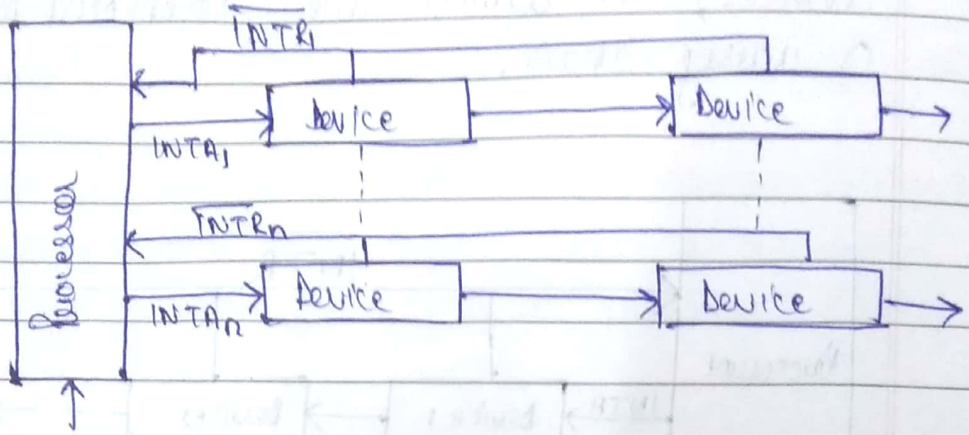


Daisy Chain

The interrupt-request line $\overline{INTR}$ is common to all devices. The interrupt-acknowledge line INTA is connected in a daisy-chain fashion, such that the INTA signal propagates serially through the devices. When several devices raise an interrupt request and the $\overline{INTR}$ line is activated, the processor responds by setting the INTA line to 1. This signal is received by device 1. Device 1 passes the signal on to device 2 only if it does not require any service. If device 1 has a pending request for interrupt, it blocks the INTA signal and proceeds to put its identifying code on the data lines.

Therefore, in daisy chain arrangement, the device that is electrically closest to the processor

has the highest priority.



Priority Arbitration Circuit

Devices are connected at a different priority level

Devices are ~~connected~~ organised in groups, and each group is connected at a different priority level. Within a group, devices are connected in a daisy chain.

* Controlling Device Request

The control needed is usually provided in the form of an interrupt-enable bit in the device's interface circuit.

There are two independent mechanisms for controlling interrupt requests. At the device end, an interrupt enable bit in a control register determines whether the device is allowed to generate an interrupt request.

At the processor end, either an interrupt enable bit in a ~~control~~ PS register or a priority structure determines whether a given interrupt will be accepted.

* Exceptions:
Exception is used to refer to any event that causes an interruption.

Recovery from Errors: Many computers contain an error checking code in their main memory, which allows detection of error in the stored data. If an error occurs, the control hardware detects it and informs the processor by raising an interrupt.
The processor may also interrupt a program if it detects an error or an unusual condition while executing the program. (eg: division by zero)
When exception processing is initiated as a result of such errors, the processor suspends the program being executed and starts an exception-service routine.

Debugging: The debugger uses exceptions to provide two important facilities called trace and breakpoints.
When a processor is operating in the trace mode, an exception occurs after execution of every instruction, using the debugger program as

exception-service routine. The debugging program enables the user to examine the contents of registers, memory locations and so on.

The trace exception is disabled during the execution of the debugging program.

Breakpoints provide a similar facility, except that the program is interrupted at specific points selected by the user. An instruction called Trap or Software-Interrupt is usually provided for this purpose.

Privilage exception: To protect the operating system of a computer from being corrupted by user programs, certain instructions can be executed only when the processor is in supervisor mode. These instructions are called privileged instructions. (eg: changing the priority level of the processor). An attempt to execute such an instruction will produce a privilege exception, causing the processor to switch to the supervisor mode and begin executing an appropriate routine.