CS501 Final Term Subjective Notes by Saeed

Memory and I/O subsystems differ in the following ways:

1. Wider range of data transfer speed:

I/O devices can be very slow such as a keyboard in which case the interval between two successive bytes (or keystrokes) can be in seconds. On the other extreme, I/O devices can be very fast such as a disk drive sending data to the CPU or a stream of packets arriving over a network,

2. Asynchronous activity:

Memory subsystems are almost always synchronous. This means that most memory transfers are governed by the CPU's clock. Generally this is not the case with I/O subsystems. Additional signals, called ***handshaking signals***, are needed to take care of asynchronous I/O transfers.

3. Larger degradation in data quality:

Data transferred by I/O subsystems can carry more noise. As an example, telephone line noise can become part of the data transferred by a modem.

4. Mechanical nature of many I/O devices:

Many I/O devices or a large portion of I/O devices use mechanical parts which inherently have a high failure rate. In case an I/O device fails, interruptions in data transfer will occur, ***reducing the throughput***. As an example, if a printer runs out of paper, then additional bytes cannot be sent to it. The CPU's data should be buffered (or kept in a temporary place) till the paper is supplied to the printer, otherwise the CPU will not be able to do anything else during this time. *To deal with these differences, special software programs called device drivers are made a part of the operating system.*

Major components of an I/O subsystem

I/O subsystems have two major parts:

• ***The I/O interface***, which is the electronic circuitry that connects the CPU to the I/O device.

• ***Peripherals***, which are the devices used to communicate with the CPU, for example, the keyboard, the monitor, etc.

Computer Interface

A Computer Interface is a piece of hardware whose primary purpose is to connect together any of the following types of computer elements in such a way *that the signal levels and the timing requirements of the elements are matched by the interface.* Those elements are:

• ***The processor unit***

• ***The memory subsystem(s)***

• ***Peripheral (or I/O) devices***

• ***The buses (also called "links")***

I/O ports serve the following three purposes:

• Buffering (i.e., holding temporarily) the data to and from the computer bus.

• Holding control information that dictates how a transfer is to be conducted.

• Holding status information so that the processor can monitor the activity of the interface and its associated I/O element.

In most cases, the word ***buffer*** refers to I/O registers in an interface where data, status or control information is temporarily stored. A block of memory locations within the main memory or within the peripheral devices is also called a buffer if it is used for temporary storage. Special circuits used in the interfaces for voltage/current matching, at the input and the output, are also called buffers.

The system bus actually consists of three buses, namely the address bus, the data bus and the control bus.

Memory Mapped I/O versus Isolated I/O

In ***isolated I/O***, a separate address space of the CPU is reserved for I/O operations. This address space is totally different from the address space used for memory devices. In other words, a CPU has two distinct address spaces, one for memory and one for input/output. Unique CPU instructions are associated with the I/O space, which means that if those instructions are executing on the CPU, then the accessed address space will be the I/O space and hence the devices mapped on the I/O space. In some processors, like the SRC, there is no separate I/O space. In this case, some address space out of the memory address space must be used to map I/O devices. The benefit will be that all the instructions which access memory can be used for I/O devices. There is no need for including separate I/O instructions in the ISA of the processor. However, the disadvantage will be that the I/O interface will become complex. If partial decoding is used to reduce the complexity of the I/O interface, then a lot of memory addresses will be consumed.

Considerations during I/O Subsystem Design

Certain things must be taken care of during the design of an I/O subsystem.

Data location:

The designer must identify the device where the data to be accessed is available, the address of this device and how to collect the data from this device.

Data transfer:

This includes the direction of transfer of data; whether it is out of the CPU or into the CPU, whether the data is being sent to the monitor or the hard drive, or whether it is being received from the keyboard or the mouse. It also includes the amount of data to be transferred and the rate at which it should be transferred

Data synchronization:

This means that the CPU should input data from an input device only when the device is ready to provide data and send data to an output device only when it is ready to receive data.

There are three basic schemes which can be used for synchronization of an I/O data transmission:

1. ***Synchronous transmission***
2. ***Semi-synchronous transmission***
3. ***Asynchronous transmission***

Synchronous transmission:

Handshaking signals are needed. At the start of the transfer operation, the master activates the Read signal, which indicates to the slave that it should respond with data. The data is provided by the slave, and the master uses the Enable signal to latch it. All activity takes place synchronously with the system clock. *A familiar example of synchronous transfer is a register-to-register transfer within a CPU.*

Semi-synchronous transmission:

All activity is still synchronous with the system clock, but in some situations, the slave device may not be able to provide the data to the master within the allotted time. The additional time needed by the slave, can be provided by adding an integral number of clock periods to Figure A the master's cycle time.

Asynchronous transmission:

This type of transfer does not require a common clock. The master and the slave operate at different speeds. *Handshaking signals are necessary in this case*, and are used to coordinate the data transfer between the master and the slave as shown in the Figure C. When the master wants to initiate a data transfer, it activates its Ready signal. The slave detects this signal, and if it can provide data to the master, it does so and also activates its Acknowledge signal. Upon receiving the Acknowledge signal, the master uses the Enable signal to latch the incoming data .The master then deactivates its Ready line, and in response to it, the slave removes its data and deactivates its Acknowledge line.

There are two ways in which data can be transferred between the CPU and an I/O device:

1. ***Serial***
2. ***Parallel***

Serial Transfer, or serial communication of data between the CPU and the I/O devices, refers to the situation when all the data bits in a "piece of information", (which is a byte or word mostly), are transferred one bit at a time, over a single pair of wires.

Advantages:

• Easy to implement, especially by using UARTs7 or USARTs8.

• Low cost because of less wires.

• Longer distance between transmitter and receiver.

Disadvantages:

• Slow by its very nature.

• Inefficient because of the associated overhead

Parallel Transfer, or parallel communication of data between the CPU and the I/O devices, refers to the situation when all the bits of data (8 or 16 usually), are transferred over separate lines simultaneously, or in parallel.

Advantages:

• Fast (compared to serial communication)

Disadvantages:

• High cost (because of more lines).

• Cost increases with distance.

• Possibility of interference (noise) increases with distance.

Remember that the terms "serial" and "parallel" are with respect to the computer I/O ports and not with respect to the CPU. The CPU always transfers data in parallel.

Types of serial communication

There are two types of serial communication:

Asynchronous:

• Special bit patterns separate the characters.

• "Dead time" between characters can be of any length.

• Clocks at both ends need not have the same frequency (within permissible limits).

Synchronous:

• Characters are sent back to back.

• Must include special "sync" characters at the beginning of each message.

• Must have special "idle" characters in the data stream to fill up the time when no information is being sent.

• Characters must be precisely spaced.

• Activity at both ends must be coordinated by a single clock. (This implies that the clock must be transmitted with data).

The "maximum information rate" of a synchronous line is higher than that of an asynchronous line with the same "bit rate", because the asynchronous transmission must use extra bits with each character.

Error conditions related to serial communication

(Some related to synchronous transmission, some to asynchronous, and some to both).

• **Framing Error:** is said to occur when a 0 is received instead of a stop bit (which is always a 1). It means that after the detection of the beginning of a character with a start bit, the appropriate number of stop bits was not detected.

• **Parity Error:** is said to occur when the parity* of the received data is not the same as it should be. [B] (PARITY is equivalent to the number of 1's; it is either EVEN or ODD. A PARITY BIT is an extra bit added to the data, for the purpose of error detection and correction. If even parity is used, the parity bit is set so that the total number of 1's, including the parity bit, is even. The same applies to odd parity.)

• **Overrun Error:** means that the prior character that was received, was not yet read from the USART's "receive data register" by the CPU, and is overwritten by the new received character. Thus the first character was lost, and should be retransmitted.

• **Under-run Error:** If a character is not available at the beginning of an interval, an underrun is said to occur. The transmitter will insert an idle character till the end of the interval.

I/O Buses

Another organization that is used in modern Computer has a memory bus for connecting the CPU to the memory subsystem. This bus is separate from the I/O bus that is used to connect peripherals and I/O devices to the system. *Examples of I/O buses include the PCI bus and the ISA bus.*

However, with the passage of time, FIGURE 1 computer architects drifted towards separate memory and I/O buses, thereby giving more flexibility to users wanting to upgrade their existing systems. A main disadvantage of I/O buses (and the buses in general) is that every bus has a fixed bandwidth which is shared by all devices on the bus. Additionally, electrical constraints like transmission line effects and bus length further reduce the bandwidth.

Bus arbitration:

Arbitration is another issue in the use of I/O buses. Most commercially available I/O buses have protocols defining a number of things, for example how many devices can access the bus, what will happen if multiple devices want to access the bus at the same time, etc. In such situations, an "arbitration scheme" must be established. As an example, in the SCSI11 specifications, every device in the system is assigned an ID which identifies the device to the "bus arbiter". If multiple devices send a request for the bus, the device with the highest priority will be given access to the bus first. *Such a scheme is easy to implement because the arbiter can easily decide which device should be given access to the bus, but its disadvantage is that the device with a low priority will not be able to get access to the bus12*. An alternate scheme would be to give the highest priority to the device that has been waiting for the longest time for the bus. As a result of this arbitration, the access time, or the latency, of such buses will be further reduced.

Thus, there are two important functions which should be built into I/O ports.

1. Address decoding

2. Data isolation for input ports or data capturing for output ports.

1. Address decoding: Since every I/O port has a unique identifier associated with it, (which is called its address, and no other port in the system should have the same address), by monitoring the system address bus, the I/O port knows when it is its turn to participate in a transfer.

2. Data isolation or capturing: For input ports, the incoming data should be placed on the data bus only during the I/O read bus cycle. At all other times, this data should be isolated from the data bus otherwise it will cause "bus contention". *Tri-state buffers are used for this purpose.* Their input lines are connected to the peripheral device supplying data and their output lines are connected to the data bus.

The NUXI Problem

It can be easily understood from the previous example that the big-endian format results in the least significant byte being transferred over the most significant side of the data bus, and vice versa. The situation will be exactly opposite when the little-endian format is used. In this case, the least significant byte will be transferred over the least side of the data bus. Now imagine a computer using the little-endian format exchanging data with a computer using the big-endian format over a 16-bit parallel port. (this may be the case when we have a network of different types of computer, for example). The data transmitted by one will be received in a "swapped" form by the other, eg., the string "UN" will be received as "NU" and the string "IX" will be received as "XI". So UNIX changes to NUXI --- hence the name NUXI problem. *Special software is used to resolve this problem.* ***Partial decoding and the "wrap around" effect***

Partial decoding is a technique in which some of the CPU's address lines forming an input to the address decoder are ignored. This reduces the complexity of the address decoder,

and also lowers the cost.

Data bus multiplexing

Data bus multiplexing refers to the situation when one part of the data bus is connected to the peripheral's data bus at one time and the second part of the data bus is connected to the peripheral's data bus at a different time in such a way that at one time, only one 8-bit portion of the data bus is connected to the peripheral.

The printer cannot read data due to one of the following reasons:

1. During data entry
2. During data printing
3. In offline state
4. During printer error status

Note#2

When the printer is in one of the following states:

1. Paper end state
2. Offline state
3. Error state

I/O techniques:

There are three main techniques using which a CPU can exchange data with a peripheral device, namely

- ***Programmed I/O***
- ***Interrupt driven I/O***
- ***Direct Memory Access (DMA).***

Programmed Input/Output

Programmed I/O refers to the situation when all I/O operations are performed *under the direct control of a program running on the CPU*. This program, which usually consists of a "**tight loop**", controls all I/O activity, including device status sensing, issuing read or write commands, and transferring the data18. A subsequent I/O operation cannot begin until the

current I/O operation to a certain device is complete. This causes the CPU to wait, and thus makes the scheme extremely inefficient.

Interrupt Driven I/O:

An interrupt is a request to the CPU to suspend normal processing and temporarily divert the flow of control through a new program. This new program to which control is transferred is called an Interrupt Service Routine or ISR. Another name for an ISR is an Interrupt Handler.

• *Interrupts are used to demand attention from the CPU.*

• *Interrupts are asynchronous breaks in program flow that occur as a result of events outside the running program.*

• *Interrupts are usually hardware related, stemming from events such as a key or button press, timer expiration, or completion of a data transfer.*

Advantages of interrupts:

• Useful for interfacing I/O devices with low data transfer rates.

• CPU is not tied up in a tight loop for polling the I/O device.

Types of Interrupts:

The general categories of interrupts are as follows:

1. ***Internal Interrupts***
2. ***External Interrupts***
   - *Hardware Interrupts*
   - *Software Interrupts*

Internal Interrupts:

• Internal interrupts are generated by the processor.

• These are used by processor to handle the exceptions generated during instruction execution. Internal interrupts are generated to handle conditions such as stack overflow or a divide-by-zero exception. Internal interrupts are also referred to as **traps**. They are mostly used for exception handling.

External Interrupts:

External interrupts are generated by the devices other than the processor. They are of two types.

• *Hardware interrupts are generated by the external hardware.*

• *Software interrupts are generated by the software using some interrupt instruction.*

As the name implies, external interrupts are generated by devices external to the CPU, such as the click of a mouse or pressing a key on a keyboard. In most cases, input from external sources requires immediate attention. These events require a quick service by the software

Hardware interrupts:

Hardware interrupts are generated by external events specific to peripheral devices. Most processors have at least one line dedicated to interrupt requests. When a device signals on this specific line, the processor halts its activity and executes an interrupt service routine. Such interrupts are always asynchronous with respect to instruction execution, and are not associated with any particular instruction.

There are two types of hardware interrupt:

1. Maskable Interrupts
2. Non-maskable Interrupts

Maskable Interrupts:

• These interrupts are applied to the INTR pin of the processor.

• These can be blocked by resetting the flag bit for the interrupts.

Non-maskable Interrupts:

• These interrupts are detected using the NMI pin of the processor.

• These can not be blocked or masked.

• Reserved for catastrophic event in the system.

Software interrupts:

Software interrupts are usually associated with the software. A simple output operation in a multitasking system requires software interrupts to be generated so that the processor may temporarily halt its activity and place the data on its data bus for the peripheral device. Output is usually handled by interrupts so that it appears interactive and asynchronous.

Interrupt Service Routine (ISR):

• It is a routine which is executed when an interrupt occurs.

• Also known as an Interrupt Handler.

• Deals with low-level events in the hardware of a computer system, like a tick of a real-time clock.

As it was mentioned earlier, an interrupt once generated must be serviced through an interrupt service routine. These routines are stored in the system memory ready for execution. Once the interrupt is generated, the processor must branch to the location of the appropriate service routine to execute it.

Branch Address of the ISR:

There are two ways used to choose the branch address of an Interrupt Service Routine.

1. Non-vectored Interrupts
2. Vectored Interrupts

Non-vectored Interrupts:

In non-vectored interrupts, the branch address of the interrupt service routine is fixed. The code for the ISR is loaded at fixed memory location. Non-vectored interrupts are very easy to implement and not flexible at all. In this case, the number of peripheral devices is fixed and may not be increased. Once the interrupt is generated the processor queries each peripheral device to find out which device generated the interrupt. This approach is the least flexible for software interrupt handling.

Vectored Interrupts:

Interrupt vectors are used to specify the address of the interrupt service routine. The code for ISR can be loaded anywhere in the memory. This approach is much more flexible as the programmer may easily locate the interrupt vector and change its addresses to use custom interrupt servicing routines. Using vectored interrupts, multiple devices may share the same interrupt input line to the processor. *A process called daisy chaining is then used to locate the interrupting device.*

Interrupt Vector:

Interrupt vector is a fixed size structure that stores the address of the first instruction of the ISR.

Interrupt Vector Table:

• All of the interrupt vectors are stored in the memory in a special table called Interrupt Vector Table.

• Interrupt Vector Table is loaded at the memory location 0 for the 8086/8088.

Interrupts in Intel 8086/8088:

• Interrupts in 8086/8088 are vector interrupts.

• Interrupt vector is of 4 bytes to store IP and CS.

• Interrupt vector table is loaded at address 0 of main memory.

• There is provision of 256 interrupts.

Branch Address Calculation:

• The number of interrupt is the number of interrupt vector in the interrupt vector table.

• Since size of each vector is 4 bytes and interrupt vector starts from address 0, therefore, the address of interrupt vector can be calculated by simply multiplying the number by 4.

Returning from the ISR:

Every ISA should have an instruction, like the IRET instruction, which should be executed when the ISR terminates. This means that the IRET instruction should be the last instruction of every ISR. This is, in effect, a FAR RETURN in that it restores a number of registers, and flags to their value before the ISR was called. Thus the previous environment is restored after the servicing of the interrupt is completed.

Interrupt Handling:

The CPU responds to the interrupt request by completing the current instruction, and then storing the return address from PC into a memory stack. Then the CPU branches to the ISR that processes the requested operation of data transfer. In general, the following sequence takes place.

Hardware Interrupt Handling:

• Hardware issues interrupt signal to the CPU.

• CPU completes the execution of current instruction. CPU acknowledges interrupt.

• Hardware places the interrupt number on the data bus.

• CPU determines the address of ISR from the interrupt number available on the data bus.

• CPU pushes the program status word (flags) on the stack along with the current value of program counter.

• The CPU starts executing the ISR.

• After completion of the ISR, the environment is restored; control is transferred back to the main program.

Interrupt Latency:

Interrupt Latency is the time needed by the CPU to recognize (not service) an interrupt request. It consists of the time to perform the following:

• Finish executing the current instruction.

• Perform interrupt-acknowledge bus cycles.

• Temporarily save the current environment.

• Calculate the IVT address and transfer control to the ISR.

Interrupt Precedence:

Interrupts occurring at the same time i.e. within the same instruction are serviced according to a pre-defined priority.

• In general, all internal interrupts have priority over all external interrupts; the single-step interrupt is an exception.

• NMI has priority over INTR if both occur simultaneously.

• The above mentioned priority structure is applicable as far as the recognition of (simultaneous) interrupts is concerned. As far as servicing (execution of the related ISR) is concerned, the single-step interrupt always gets the highest priority, then the NMI, and finally those (hardware or software) interrupts that occur last. If IF is not 1, then INTR is ignored in any case. Moreover, since any ISR will clear IF, INTR has lower "service priority" compared to software interrupts, unless the ISR itself sets IF=1.

Simultaneous Hardware Interrupt Requests:

The priority of the devices requesting service at the same time is resolved by using two ways:

1. Daisy-Chained Interrupt
2. Parallel Priority Interrupt

Daisy-Chaining Priority:

• The daisy-chaining method to resolve the priority consists of a series connection of the devices in order of their priority.

• Device with maximum priority is placed first and device with least priority is placed at the end.

Daisy-Chain Priority Interrupt

• The devices interrupt the CPU.

• The CPU sends acknowledgement to the maximum priority device.

• If the interrupt was generated by the device, the interrupt for the device is serviced.

• Otherwise the acknowledgement is passed to the next device.

Parallel Priority:

• Parallel priority method for resolving the priority uses individual bits of a priority encoder.

• The priority of the device is determined by position of the input of the encoder used for the interrupt.

Parallel Priority Interrupt:

Comparison of Interrupt driven I/O and Polling

Interrupt driven I/O is better than polling. In the case of polling a lot of time is wasted in questioning the peripheral device whether it is ready for delivering the data or not. In the case of interrupt driven I/O the CPU time in polling is saved.

Design Issues

There are two design issues:

1. Device Identification
2. Priority mechanism

Device Identification

In this issue different mechanisms could be used.

• Multiple interrupt lines

• Software Poll

• Daisy Chain

1. **Multiple Interrupt Line**

This is the most straight forward approach, and in this method, a number of interrupt lines are provided between the CPU and the I/O module. However, it is impractical to dedicate more than a few bus lines or CPU pins to interrupt lines. Consequently, even if multiple lines are used, it is likely that each line will have multiple I/O modules attached to it. Thus on each line, one of the other technique would still be required.

2. Software Poll

CPU polls to identify the interrupting module and branches to an interrupt service routine on detecting an interrupt. This identification is done using special commands or reading the device status register. Special command may be a test I/O.

3. Daisy Chain

The wired or interrupt signal allows several devices to request interrupt simultaneously. However, for proper operation one and only one requesting device must receive an acknowledge signal, otherwise if we have more than one devices, we would have a data bus contention and the interrupt information would not be resolved. The usual solution is called a daisy chain.

Disadvantages of Software Poll and Daisy Chain

The software poll has a disadvantage is that it consumes a lot of time, while the daisy chain is more efficient. The daisy chain has the disadvantage that the device nearest to the CPU would have highest priority. So, usually those devices which require higher priority would be connected nearer to the CPU. Now in order to get a fair chance for other devices, other mechanisms could be initiated or we could say that we could start instead of device 0 from that device where the CPU finishes the last interrupt and could have a cyclic provision to different devices. To make things simple, we have made the assumption that only one interrupt pin is available on the FALCON-A, and only one interrupt is possible at a given time with this CPU. Implications of allowing only one interrupt at a time are that

• No NMI is possible

• No nesting of interrupts is possible

• No priority structure needed for multiple devices

• No arbitration needed for simultaneous interrupts

• No need for vectored interrupts, therefore, no need of interrupt vectors and interrupt vector tables

• Effect of software initiated interrupts and internal interrupts (exceptions) has to be ignored in this discussion

Along with the previous assumption, the following assumptions have also been used:

• Hardware sets and clears the interrupt flag, in addition to handling other things like saving PC, etc.

• The address of the ISR is stored at absolute address 2 in memory.

• The ISR will set up a stack in the memory for saving the CPU's environment

• One ASCII character stored per 16-bit word in the FALCON-A's memory and one character transferred during a 16-bit transfer.

• The calling program will call the ISR for printing the first character through the printer driver.

• Printer will activate ACKNLG# only when not BUSY.

Interrupt Software:

Our software for the interrupt driven printer example consists of three parts:

1). Dummy calling program

2). Printer Driver

3). ISR

The following activities are also performed by the driver

• Set port addresses

• Set up variables for the STROBE# puls

• Initialize printer and enable its IRQEN.

• Set up printer ISR by pointing to the buffer and initializing counter

• Make sure that the previous print job is not in progress

• Set PB flag to block further print jobs till current one is complete

• Invoke ISR for the first time

• Pass error message to main program if ISR reports an error

• Return to main program

Falsim

Direct Memory Access (DMA)

Direct memory access is a technique, where by the CPU passes its control to the memory subsystem or one of its peripherals, so that a contiguous block of data could be transferred from peripheral device to memory subsystem or from memory subsystem to peripheral device or from one peripheral device to another peripheral device.

Advantage of DMA

The transfer rate is pretty fast and conceptually you could imagine that through disabling the tristate buffers, the system bus is isolated and a direct connection is established between the I/O subsystem and the memory subsystem and then the CPU is free. It is idle at that time or it could do some other activity. Therefore, the DMA would be quite useful, if a large amount of data needs to be transferred, for example from a hard disk to a printer or we could fill up the buffer of a printer in a pretty short time.

As compared to interrupt driven I/O or the programmed I/O, DMA would be much faster.

What is the consequence? The consequence is that we need to have another chip, which is a ***DMA controller***. "A DMA controller could be a CPU in itself and it could control the total activity and synchronize the transfer of data". DMA could be considered as a technique of transferring data from I/O to memory and from memory to I/O without the intervention of the CPU. The CPU just sets up an I/O module or a memory subsystem, so that it passes control and the data could be passed on from I/O to memory or from memory to I/O or within the memory from one subsystem to another subsystem without interaction of the CPU. After this data transfer is complete, the control is passed from I/O back to the CPU.

DMA Approach

The DMA approach is to turn off i.e. through tri-state buffers and therefore, electrically disconnect from the system bus, the CPU and let a peripheral device or a memory subsystem or any other module or another block of the same module communicate directly with the memory or with another peripheral device. This would have the advantage of having higher transfer rates which could approach that of limited by the memory itself.

Disadvantage of DMA

The disadvantage however, would be that an additional DMA controller would be required, that could make the system a bit more complex and expensive. Generally, the DMA requests have priority over all other bus activities including interrupts. No interrupts may be recognized during a DMA cycle.

The following DMA transfer combinations are possible:

- ***Memory to memory***
- ***Memory to peripheral***
- ***Peripheral to memory***
- ***Peripheral to peripheral***

Thus, the real cause of the limited transfer rate is the CPU itself. It acts as an unnecessary "middleman". The above discussion also implies that, in general, every data word travels over the system bus twice.

Some Definitions:

• **MASTER COMPONENT**: A component connected to the system bus and having control of it during a particular bus cycle.

• **SLAVE COMPONENT**: A component connected to the system bus and with which the master component can communicate during a particular bus cycle. Normally the CPU with its bus control logic is the master component.

• **QUALIFICATIONS TO BECOME A MASTER:** A Master must have the capability to place addresses on the address bus and direct the bus activity during a bus cycle.

• **QUALIFIED COMPONENTS:** Processors with their associated bus control logic.

o DMA controllers.

• **CYCLE STEALING:** Taking control of the system bus for a few bus cycles.

Data Transfer using DMA:

Data transfer using DMA takes place in three steps.

1st Step:

In this step when the processor has to transfer data it issues a command to the DMA controller with the following information:

Operation to be performed i.e., read or write operation. Address of I/O device.

Address of memory block.

Size of data to be transferred.

After this, the processor becomes free and it may be able to perform other tasks.

2nd Step:

In this step the entire block of data is transferred directly to or from memory by the DMA controller.

3rd Step:

In this, at the end of the transfer, the DMA controller informs the processor by sending an interrupt signal.

DMA Configurations:

1. Single Bus Detached DMA
2. Single Bus Integrated DMA
3. I/O Bus

Single Bus Detached DMA

In the example provided by the above diagram, there is a single bidirectional bus connecting the processor, the memory, the DMA module and all the I/O modules. When a particular I/O module needs to read or write large amounts contiguous data it requests the processor for direct memory access. If permission is granted by the processor, the I/O module sends the read or write address and the size of data needed to be read or written to the DMA module. Once the DMA module acknowledges the request, the I/O module is free to read or write its contiguous block of data from or onto main memory.

Even though in this situation the processor will not be able to execute while the transfer is going on (as there is a just a single bus to facilitate transfer of data), DMA transfer is much faster than having each word of memory being read by the processor and then being written to its location.

Single Bus Integrated DMA

In this configuration the DMA and one or more I/O modules are integrated without the inclusion of system bus functioning as the part of I/O module or may be as a separate module controlling the I/O module.

IO Bus

In this configuration we integrate the DMA and I/O modules through an I/O bus. So it will cut the number of I/O interfaces required between DMA and I/O module.

Cycle Stealing

The DMA module takes control of the bus to transfer data to and from memory by forcing the CPU to temporarily suspend its operation. This approach is called Cycle Stealing because in this approach DMA steals a bus cycle.

DMA and Interrupt breakpoints during an instruction cycle

The CPU suspends or pauses for one bus cycle when it needs a bus cycle, transfers the data and then returns the control back to the CPU.

I/O processors

When I/O module has its own local memory to control a large number of I/O devices without the involvement of CPU is called I/O processor.

I/O Channels

When an I/O module has a capability of executing a specific set of instructions for specific I/O devices in the memory without the involvement of CPU is called I/O channel.

I /O channel architecture:

Types of I/O channels:

Selector Channel

It is the DMA controller that can do block transfers for several devices but only one at a time.

Multiplexer Channel

It is the DMA controller that can do block transfers for several devices at once.

Types of Multiplexer Channel

1. Byte Multiplexer
2. Block Multiplexer

Byte Multiplexer

• Byte multiplexer accepts or transmits characters.

• Interleaves bytes from several devices.

• Used for low speed devices.

Block Multiplexer

• Block multiplexer accepts or transmits block of characters.

• Interleaves blocks of bytes from several devices.

• Used for high speed devices.

Virtual Address:

Virtual address is generated be the logical by the memory management unit for translation.

Physical Address:

Physical address is the address in the memory.

DMA and memory system

DMA disturbs the relationship between the memory system and CPU.

Direct memory access and the memory system Without DMA, all memory accesses are handled by the CPU, using address translation and cache mechanism. When DMA is implemented into an I/O system memory accesses can be made without intervening the CPU for address translation and cache access. *The problems created by the DMA in virtual memory and cache systems can be solved using hardware and software techniques.*

One solution to the problem is that all the I/O transfers are made through the cache to ensure that modified data are read and updated in the cache on the I/O write. This method can decrease the processor performance because of infrequent usage of the I/O data.

Another approach is that the cache is invalidated for an I/O read and for an I/O write, write-back (flushing) is forced by the operating system. This method is more efficient because flushing of large parts of cache data is only done on DMA block accesses.

Third technique is to flush the cache entries using a hardware mechanism, used in multiprogramming system to keep cache coherent.

Hard Disk

Peripheral devices connect the outside world with the central processing unit through the I/O modules. One important feature of these peripheral devices is the variable data rate. Peripheral devices are important because of the function they perform.

A hard disk is the most frequently used peripheral device. It consists of a set of **platters**. Each platter is divided into **tracks**. The track is subdivided into **sectors**. To identify each sector, we need to have an address. So, before the actual data, there is a header and this header consisting of few bytes like 10 bytes. Along with header there is a trailer. *Every sector has three parts: a header, data section and a trailer.*

Static Properties

The storage capacity can be determined from the number of platters and the number of tracks. In order to keep the density same for the entire surface, the trend is to use more number of sectors for outer tracks and lesser number of sectors for inner tracks.

Dynamic Properties

When it is required to read data from a particular location of the disk, the head moves towards the selected track and this process is called ***seek***. The disk is constantly rotating at a fixed speed. After a short time, the selected sector moved under the head. This interval is called the ***rotational delay***. On the average, the data may be available after half a revolution. Therefore, the rotational latency is ***half revolution***.

The time required to seek a particular track is defined by the manufacturer. Maximum, minimum and average seek times are specified. Seek time depends upon the present position of the head and the position of the required sector. For the sake of calculations, we will use the average value of the seek time.

• **Transfer rate**

When a particular sector is found, the data is transferred to an I/O module. This would depend on the transfer rate. It would typically be between 30 and 60 Mbytes/sec defined by the manufacturer.

• Overhead time

Up till now, we have assumed that when a request is made by the CPU to read data, then hard disk is available. But this may not be the case. In such situation we have to face a queuing delay. There is also another important factor: the hard disk controller, which is the electronics present in the form of a printed circuit board on the hard disk. So the time taken by this controller is called over head time.

Mechanical Delay and Flash Memory

Mechanical movement is involved in data transfer and causes mechanical delays which are not desirable in embedded systems. To overcome this problem in embedded systems, flash memory is used. *Flash memory can be thought of a type of electrically erasable PROM. Each cell consists of two MOSFET and in between these two transistors, we have a control gate and the presence/absence of charge tells us that it is a zero or one in that location of memory.*

The basic idea is to reduce the control overheads, and for a FLASH chip, this control overhead is low. Furthermore flash memory has low power dissipation. For embedded devices, flash is a better choice as compared to hard disk. Another important feature is that read time is small for flash. However the write time may be significant. The reason is that we first have to erase the memory and then write it. However in embedded system, number of write operations is less so flash is still a good choice.

Semiconductor Memory vs. Hard Disk

At one time developers thought that development of semiconductor memory would completely wipe out the hard disk. There are two important features that need to be kept in mind in this regard:

1. Cost

It is low for hard disk as compared to semi-conductor memory.

2. Latency

Typically latency of a hard disk is in milliseconds. For SRAM, it is 105 times lower as compared to hard disk.

Error Control

There are two main issues in error control:

1. Detection of Error
2. Correction of Error

For detection of error, we just need to know that there exists an error. When the error is detected then the next step is to ask the source to resend that information. This process is called automatic request for repeat. In some cases there is also possibility that redundancy is enough and we reconstruct and find out exactly which particular bits are in error. This is called error correction.

There are three schemes commonly used for error control.

1. Parity code
2. Hamming code
3. CRC mechanism

1. Parity code

Along with the information bits, we add up another bit, which is called the parity bit. The objective is the total number of 1's as even or odd. If the parity at the receiving end is different, an error is indicated. Once error is found, CPU may request to repeat that data. The concept of parity bit could be enhanced. In such a case, we would like to increase the distance between different code words. Consider a code word consists of four bits, 0000, and second code word consists of 1111. The distance between two codes is four. So the distance between the two codes would be the number of bits in which they differ from each other. So the concept of introducing redundancy is increase this distance. Larger the distance, higher will be the capacity of the code. For single parity, the distance is two, we can only detect the parity. But if the distance is three, we could also correct these single errors.

2. Hamming code

Hamming code is an example of block code. We have an encoder which could be a program or a hardware device. We feed k inputs to it. These are k information input bits. We also feed some extra bits. Let r be the number of redundant bits. So at output we have r+k = m bits. As an example, for parity bit, we have k=7 and r=1 and m=8. So for 7 bits we get eight output bits.

For any positive integer m<=3, a Hamming code with following parameters exists:

• Code Length: n=2m-1

• Number of information symbols: k = 2m-1-m

• Number of parity-check symbols: n – k = m

3. **CRC**

The basic principle for CRC is very simple. We divide a particular code word and make it divisible by a prime number, and if it is divisible by a prime number then it is a valid code word. CRC does not support error correction but the CRC bits generated can be used to detect multi-bit errors. At the transmitter, we generate extra CRC bits, which are appended to the data word and sent along. The receiving entity can check for errors by re computing the CRC and comparing it with the one that was transmitted. CRC has lesser overhead as compared to Hamming code. It is practically quite simple to implement and easy to use.

RAID

The main advantage of having an array of disks is that we could have a simultaneous I/O request.

Latency could also be reduced...

RAID Level 0

• Not a true member of the RAID family.

• Does not include redundancy to improve performance.

• In few applications, capacity and performance are primary concerns than improved reliability. So RAID level 0 is used in such applications.

• The user and system data are distributed across all the disks in the array.

• Notable advantage over the use of a single large disk.

• Two requests can be issued in parallel, reducing the I/O queuing time.

Performance of RAID Levels

Performance of RAID Levels depends upon two factors:

• Request pattern of the host system

• Layout of the data

Similarities between RAID Levels 2 and 3

• Make use of parallel access techniques.

• All member disks participate in execution of every request.

• Spindles of the individual drives are synchronized

• Data striping is used.

• Strips are as small as a single byte or word.

Differences between RAID2 and RAID 3

• In RAID 2, error-correcting code is calculated across corresponding bits on each data disk.

• RAID 3 requires only a single redundant disk.

• Instead of an error-correcting code, a simple parity bit is computed for the set of individual bits in RAID 3

RAID Level 4

• Make use of independent access technique.

• Data striping is used.

• A bit-by-bit parity strip is calculated across corresponding strip on each data disk.

• Involves a write penalty when an I/O write request of small size is performed.

• To calculate the new parity, the array management software must read the old user parity strip.

RAID Level 5

• Organized in a similar fashion to RAID 4

• The only difference is that RAID 5 distributes the parity strips across all disks.

Introduction to ALSU

ALSU is a combinational circuit so inside an ALSU, we have AND, OR, NOT and other

different gates combined together in different ways to perform addition, subtraction, and, or, not, etc. Up till now, we consider ALSU as a "black box" which takes two operands, a and b, at the input and has c at the output. Control signals whose values depend upon the opcode of an instruction were associated with this black box.

Representation of Numbers

There are four possibilities to represent integers.

1. Sign magnitude form
2. Radix complement form
3. Diminished radix complement form
4. Biased representation

Sign magnitude form

• This is the simplest form for representing a signed number

• A symbol representing the sign of the number is appended to the left of the number

• This representation complicates the arithmetic operations

Radix complement form

• This is the most common representation.

• Given an m-digit base b number x, the radix complement of x is xc = (bm– x) mod bm

• This representation makes the arithmetic operations much easier.

Diminished radix complement form

• The diminished radix complement of an m-digit number x is xc'=bm -1- x

• This complement is easier to compute than the radix complement.

• The two complement operations are interconvertible, as

• xc= (xc'+1)mod bm

Overflow

When two m-bit numbers are added and the result exceeds the capacity of an m-bit destination, this situation is called an overflow

The sum can be computed by the two methods:

1. Ripple Carry Adder
2. Carry Look ahead Adder

There are three methods for the multiplication of sign digits:

1. 2's complement multiplier
2. Booth recoding
3. Bit-Pair recoding

2's complement Multiplication

If numbers are represented in 2's complement form then the following three modifications are required:

1. Provision for sign extension
2. Overflow prevention
3. Subtraction as well as addition of the partial product

Booth Recoding

The Booth Algorithm makes multiplication simple to implement at hardware level and speed up the procedure. This procedure is as follows:

• Start with LSB and for each 0 of the original number, place a 0 in the recorded number until a 1 in indicated.

• Place a 1 for 1in the recorded table and skip any succeeding 1's until a 0 is encountered.

• Place a 0 with 1 and repeat the procedure.

Bit-Pair Recoding

Booth recoding may increase the number of additions due to the number of isolated 1s. To avoid

this, bit-pair recoding is used. In bit-pair recoding, bits are encoded in pairs so there are only n/2

additions instead of n.

Branch Architecture

The next important function perform by the ALU is branch. Branch architecture of a machine is based on

1. Condition Codes
2. Conditional Branches

Condition Codes

Condition Codes are computed by the ALU and stored in processor status register. The 'comparison' and 'branching' are treated as two separate operations. This approach is not used in the SRC.

Floating Point Representations

Example

-0.5×10^{-3}

Sign = -1

Significand= 0.5

Exponent= -3

Base = 10= fixed for given type of representation

Significant is also called mantissa

Normalization

A normalized, non-zero floating point number has a significand whose left-most digit is non-zero and is a single number.

Example

0.56×10^{-3}…….. (Not normalized)

5.6×10^{-3}…….. (Normalized form)

Same is the case for binary.

IEEE Floating-Point Standard

IEEE floating -point standard has the following features.

Single-Precision Binary Floating Point Representation

• 1-bit sign

• 8-bit exponent

• 23-bit fraction

• A bias of 127 is used.

Double precision Binary Floating Point Representation

• 1-bit sign

• 11-bit exponent

• 52-bit fraction

• Exponent bias is 1023

Floating –Point Addition and Subtraction

The following are the steps for floating-point addition and subtraction.

• Unpack sign , exponent and fraction fields

• Shift the significant

• Perform addition

• Normalize the sum

• Round off the result

• Check for overflow

Floating-Point Multiplication

The floating-point multiplication uses the following steps:

• Unpack sign, exponent and significands

• Apply exclusive-or operation to signs, add exponents and then multiply significands.

• Normalize, round and shift the result.

• Check the result for overflow.

• Pack the result and report exceptions.

Floating-Point Division

The floating-point division us es the following steps:

• Unpack sign, exponent and significants

• Apply exclusive-or operation to signs, subtract the exponents and then divide the significants.

• Normalize, round and shift the result.

• Check the result for overflow.

• Pack the result and report exceptions.

CPU to Memory Interface

The memory address register (MAR) is m-bits wide and contains memory address generated by the CPU directly connected to the m-bit wide address bus. The memory buffer register (MBR) is w-bit wide and contains a data word, directly connected to the data bus

which is b-bit wide. The register file is a collection of 32, 32-bit wide registers used for data transfer between memory

A memory cell provides four functions: Select, DataIn, DataOut, and Read/Write. DataIn means input and DataOut means output. The select signal would be enabled to get an operation of Read/Write from this cell.

Dynamic RAM

As an alternate to the SRAM cell, the data can be stored in the form of a charge on a capacitor (a charging/discharging transistor that can become a valid memory element), and this type of memory is called dynamic memory. The capacitor has to be refreshed and recharged to avoid data loss.

Dynamic RAM Cell Operation

In a DRAM cell, the storage capacitor will discharge in around 4-15ms. Refreshing the capacitor by reading or sensing the value on bit line, amplifying it, and placing it back on to the bit line is required. The need to refresh the DRAM cell complicates the DRAM system design.

Read Only Memory (ROM)

ROM is the read-only memory which contains permanent pattern of data that cannot be changed. ROM is nonvolatile i.e. it retains the information in it when power is removed from it. Different types of ROMs are discussed below.

PROM

The PROM stands for Programmable Read only Memory. It is also nonvolatile and may be written into only once. For PROM, the writing process is performed electrically in the field. PROMs provide flexibility and convenience.

EPROM

Erasable Programmable Read-only Memory or EPROM chips have quartz windows and by applying ultraviolet light erase the data can be erased from the EPROM. Data can be restored in an EPROM after erasure. EPROMs are more expensive than PROMs and are generally used for prototyping or small-quantity, special purpose work.

EEPROM

EEPROM stands for Electrically Erasable Programmable Read-only Memory. This is a read mostly memory that can be written into at any time without erasing prior contents; only the byte or bytes addressed are updated. The write operation takes considerably longer than the read operation. It is more expensive than EPROM.

Flash Memory

An entire flash memory can be erased in one or a few seconds, which is much faster than EPROM. In addition, it is possible to erase just blocks of memory rather than an entire chip.

Cache

Cache by definition is a place for safe storage and provides the fastest possible storage after the registers. The cache contains a copy of portions of the main memory. When the CPU attempts to read a word from memory, a check is made to determine if the word is in the cache. If so, the word is delivered to the CPU. If not, a block of the main memory, consisting of some fixed number of words, is read into the cache and then t he word is delivered to the CPU.

Spatial Locality

This would mean that in a part of a program, if we have a particular address being accessed then it is highly probable that the data available at the next address would be highly accessed.

Temporal Correlation

In this case, we say that at a particular time, if we have utilized a particular part of the memory then we might access the adjacent parts very soon.

Cache Hit and Miss

When the CPU needs some data, it communicates with the cache, and if the data is available in the cache, we say that a cache hit has occured. If the data is not available in the cache then it interacts with the main memory and fetches an appropriate block of data. This is a cache miss.

Cache Organization and Functions:

The working of the cache is based on the principle of locality which has two aspects. Spatial

Locality: refers to the fact when a given address has been referenced, the next address is highly probable to be accessed within a short period of time.

Temporal Locality refers to the fact that once a particular data item is accessed, it is likely that it will be referenced again within a short period of time.

Cache Management

To manage the working of the cache, cache control unit is implemented in hardware, which performs all the logic operations on the cache. As data is exchanged in blocks between main memory and cache, four important cache functions need to be defined.

- Block Placement Strategy
- Block Identification
- Block Replacement
- Write Strategy

Cache System consists of two components.

1. Fast Memory
2. Control Logic Unit

Control logic is further divided into two parts.

Determine and Comparison Unit: For determining and comparisons of the different parts of the address and to evaluate hit or miss.

Tag RAM: Second part consists of tag memory which stores the part of the memory address (called tag) of the information (block) placed in the data cache. It also contains additional bits used by the cache management logic.

Data Cache: is a block of fast memory which stores the copies of data and instructions frequently accessed by the CPU.

Cache Strategies

Block Placement

Block placement strategy needs to be defined to specify where blocks from main memory will be placed in the cache and how to place the blocks. Now various methods can be used to map main memory blocks onto the cache .One of these methods is the associative mapping explained below.

Associative Mapping:

In this technique, block of data from main memory can be placed at any location in the cache memory. A given block in cache is identified uniquely by its main memory block number, referred to as a tag, which is stored inside a separate tag memory in the cache. To check the validity of the cache blocks, a valid bit is stored for each cache entry, to verify whether the information in the corresponding block is valid or not. Main memory address references have two fields.

• The word field becomes a "cache address" which specifies where to find the word in the cache.

• The tag field which must be compared against every tag in the tag memory.

Direct Mapping

In this technique, a particular block of data from main memory can be placed in only one location into the cache memory. It relies on principle of locality. Cache address is composed of two fields:

• Group field

• Word field

Valid bit specifies that the information in the selected block is valid.

Advantage:

Simplicity

Disadvantage:

Only a single block from a given group is present in cache at any time. Direct map Cache imposes a considerable amount of rigidity on cache organization.

Replacement Strategy

For a cache miss, we have to replace a cache block with the data coming from main memory. Different methods can be used to select a cache block for replacement.

Always Replacement:

For Direct Mapping on a miss, there is only one block which needs replacement called always replacement.

For associative mapping, there are no unique blocks which need replacement .In this case there are two options to decide which block is to be replaced.

• Random Replacement: To randomly select the block to be replaced

• LFU: Based on the statistical results, the block which has been least used in the recent past, is replaced with a new block.

Write Strategy

When a CPU command to write to a memory data will come into cache, the writing into the cache requires writing into the main memory also.

Write Through: As the data is written into the cache, it is also written into the main memory called Write Through. The advantages are:

• Read misses never result in writes to the lower level.

• Easy to implement than write back

Write Back: Date resides in the cache, till we need to replace a particular block then the data of that particular block will be written into the memory if that needs a write, called write back. The advantages are:

• Write occurs at the speed of the cache

• Multiple writes with in the same block requires only one write to the lower memory.

• This strategy uses less memory bandwidth, since some writes do not go to the lower level; useful when using multi processors.

Dirty bit is a status bit which indicates whether the block in cache is dirty (it has been modified) or clean (not modified). If a block is clean, it is not written on a miss, since lower level contains the same information as the cache. This reduces the frequency of writing back the blocks on replacement.

Write Stalls: For write to complete in Write through, the CPU has to wait. This wait state is called write stall.

Write Buffer: reduces the write stall by permitting the processor to continue as soon as the data has been written into the buffer, thus allowing overlapping of the instruction execution with the memory update.

Write Strategy on a Cache Miss

On a cache miss, there are two options for writing.

Write Allocate: The block is loaded followed by the write. This action is similar to the read miss. It is used in write back caches, since subsequent writes to that particular block will be captured by the cache.

No Write Allocate: The block is modified in the lower level and not loaded into the cache. This method is generally used in write through caches, because subsequent writes to that block still have to go to the lower level.

Virtual Memory

Virtual memory acts as a cache between main memory and secondary memory. Data is fetched in advance from the secondary memory (hard disk) into the main memory so that data is already available in the main memory when needed. The benefit is that the large access delays in reading data from hard disk are avoided. Pages are formulated in the secondary memory and brought into the main memory. This process is managed both in hardware (Memory Management Unit) and the software (The operating systems is responsible for managing the memory resources).

Advantages of Virtual Memory

• Simplified addressing scheme: the programmer does not need to bother about the exact locations of variables/instructions in the physical memory. It is taken care of by the operating system.

• For a programmer, a large virtual memory will be available, even for a limited physical memory.

• Simplified access control.

Virtual Memory Organization

Virtual memory can be organized in different ways. This first scheme is segmentation.

Segmentation:

In segmentation, memory is divided into segments of variable sizes depending upon the requirements. Main memory segments identified by segments numbers, start at virtual address 0, regardless of where they are located in physical memory. In pure segmented systems, segments are brought into the main memory from the secondary memory when

needed. If segments are modified and not required any more, they are sent back to secondary memory.

Paging:

In this scheme, we have pages of fixed size. In demand paging, pages are available in secondary memory and are brought into the main memory when needed.

Page Table

The page table entry for each page has two fields.

Page field

Control Field: This includes the following bits.

• Access control bits: These bits are used to specify read/write, and execute permissions.

• Presence bits: Indicates the availability of page in the main memory.

• Used bits: These bits are set upon a read/ write.

Performance of I/O Subsystems

There are three methods to measure I/O subsystem performance:

• Straight away calculations using execution time

• Simulation

• Queuing Theory

Loss System

Loss system is a simple system having no buffer so it does not have any provision for the queuing. In a loss system, provision is time in term of how many switches we do need, then provide some redundancy how many individuals I/O controllers we do need, then how many CPUs are there. It is also called dimension of a loss system.

Delay System

This system provides additional facilities. If we find some call party busy, we can have provision of call waiting. If we have more than one call waiting, then once we finish the first call, we may receive the second call.

Little's Law

For a system with multiple independent requests for I/O service and input rate equal to output rate, we use Little's law to find the mean number of tasks in the system and Time

sys such that Server utilization is also called traffic intensity and its value must be between 0 and 1. Server utilization depends upon two parameters:

1. Arrival Rate
2. Average time required to serve each task

So, we can say that it depends on the I/O bandwidth and arrival rate of calls into the system.

Benchmarks programs

In order to measure the performance of real systems and to collect the values of parameters needed for prediction, Benchmark programs are used.

Types of Benchmark programs

Two types of benchmark programs are used:

TPC-C

SPEC

Asynchronous I/O and operating system

In order to improve the I/O performance, parallelism is used.

For this, two approaches are available:

• Synchronous I/O

• Asynchronous I/O

Synchronous I/O

In this approach, operating system requests data and switches to another process. Until the desire data arrived. Then the operating system switches back to the requesting process.

Asynchronous I/O

This model is of the process to continue after making a request and it is not blocked until it tries to read requested data.

Bus versus switches

Consider a LAN, using bus topology. If we replace the bus with a switch, the speed of the data transfer will be improved to a great extent.

Introduction to Computer Networks

A computer architect should know about computer networks because of the two main reasons:

Connectivity

Connection of components with in a single computer follows the same principles used for the connection of different computers. It is important for the computer architect to know about connectivity for better sharing of bandwidth

Sharing of resources

Consider a lab with 50 computers and 2 printers using a network, all these 50 computers can s hare these 2 printers.

Protocol

A set of rules followed by different components in a network. These rules may be defined for hardware and software.

Host

It is a computer with a modem, LAN card and other network interfaces. Hosts are also called nodes or end points. Each node is a combination of hardware and software and all nodes are interconnected by means of some physical media.

Difference between Distributed Computing and Computer Networks

In distributed computing, all elements which are interconnected operate under one operating system. To a user, it appears as a virtual uni-processor system.

In a computer network, the user has to specify and log in on a specific machine. Each machine on the network has a specific address. Different machines communicate by using the network which exists among them.

We can classify a network based on the following two parameters:

• The number and type of machines to be interconnected

• The distance between these machines

B ased on these two parameters, we have the following type of networks:

SAN (System/Storage Area Network)

It refers to a cluster of machines where large disk arrays are present. Typical distances could be

tens of meters.

LAN (Local Area Network)

It refers to the interconnection of machines in a building or a campus. Distances could be in Kilometers.

WAN (Wide Area Network)

It refers to the interconnection between LANs.

Interconnectivity in WAN

Two methods are used to interconnect WANs:

1. Circuit switching

It is normally used in a telephone exchange. It is not an efficient way.

2. Packet switching

A block (an appropriate number of bits) of data is called a packet. Transfer of data in the form packets through different paths in a network is called packet switching. Additional bits are usually associated with each packet. These bits contain information about the packet. These additional bits are of two types: header and trailer. As an example, a packet may have the form s hown below:

Error detection

The trailer can be used for error detection. In the above example, a 4 bit checksum can be used to detect any error in the packet. The errors in the message could be due to the long distancetransmission. If the error is found in some message, then this message will be repeated. For a reliable data transmission, bit error rate should be minimum.

Software steps for sending a message:

• Copy data to the operating system buffer.

• Calculate the checksum, include in trailer and star timer.

• Send data to the hardware for transmission.

Software steps for message reception:

• Copy data to the operating system buffer.

• Calculate the checksum; if same, send acknowledge and copy data to the user area otherwise discard the message.

Response of the sender to acknowledgment:

• If acknowledgment arrives, release copy of message from the system buffer.

• When timer expires, resend data and restart the time.

Performance Issues

1. Bandwidth

It is the maximum rate at which data could be transmitted through networks. It is measured in bits/sec.

2. Latency

In a LAN, latency (or delay) is very low, but in a WAN, it is significant and this is due to the switches, routers and other components in the network

3. Time of flight

It is the time for first bit of the message to arrive at the receiver including delays. Time of the flight increases as the distance between the two machines increases.

4. Transmission time

T he time for the message to pass through the network, not including the time of flight.

5. Transport latency

T ransport latency= time of flight + transmission time

6. Sender overhead

It is the time for the processor to inject message in to the network.

7. Receiver overhead

It is the time for the processor to pull the message from the network.

8. Total latency

T otal latency = Sender overhead + Time of flight + Message size/Bandwidth + Receiver overhead

9. Effective bandwidth

Effective bandwidth = Message size/Actual Bandwidth Actual bandwidth may be larger than the effective bandwidth.

Effective bandwidth versus Message size

Effective bandwidth is always less than the raw bandwidth. If the effective bandwidth is closer to the raw bandwidth, the size of the message will be larger. If the message size is larger then network will be more effective.

If large number of the messages are present then a queue will be formed, and the user has to face delay. To minimize the delay, it is better to use packets of small size.

Modem

To interconnect different computers by using twisted pair copper wire, an interface is used which is called modem. Modem stands for modulation/demodulation. Modems are very useful to utilize the telephone network (i.e. 4 KHz bandwidth) for data and voice transmission.

Quality of Telephone Line

Data transfer rate depends upon the quality of telephone line. If telephone line is of fine quality, then data transfer rate will be sufficiently high. If the phone line is noisy then data transfer rate will be decreased.

Classification of Fiber Optic Cables

Fiber optic cables can be classified into the following types.

Multimode fiber

This fiber has large diameter. When light is injected, it disperses, so the effective data rate decreases.

Mono mode Fiber

Its diameter is very small. So dispersion is small and data rate is very high.

Wavelength –Division Multiplexing (WDM)

Waves of different wavelengths are simultaneously sent through fiber. So as a result, throughput increases.

Wireless Transmission

This is another effective medium for data transfer. Data is transferred in the form of electromagnetic waves. It has the following features:

• Data rate is in Mbits/Sec.

• Very effective because of flexibility.

• Band width is much less than fiber.

Shared/Switched Medium

Shared Medium

If a number of computers are connected with a single physical medium (i.e. coaxial or fiber), this situation is called shared medium. Because of many computers, collision takes place and affects the data transfer rate. As the number of machines on a physical medium increases, the data transfer rate decreases.

Switched Medium

To increase the throughput, a switched medium is used.

Connection Oriented vs. Connection less Communication

Connection Oriented Communication

• In this method, same path is always taken for the transfer of messages.

• It reserves the bandwidth until the transfer is complete. So no other server could use that path until it becomes free.

• Telephone exchange and circuit switching is the example of connection oriented communication.

Connection less Communication

• Here message is divided into packets with each packet having destination address.

• Each packet can take different path and reach the destination from any route by looking at its address.

• Postal system and packet switching are examples of connection less communication.

Network Topologies

Computers in a network can be connected together in different ways. The following three t opologies are commonly used:

• Bus topology

• Star topology

• Ring topology

Bus Topology

In this arrangement, computers are connected via a single shared physical medium.

Star topology

Computers are connected through a hub. All messages are broad cast because the hub is not an intelligent device.

Ring Topology

All computers are connected through a ring. Only one computer can transmit data at one time, having a pass called “Token”.

Seven Layer OSI Model

There are seven layers in this model.

1. Physical Layer
2. Data Layer
3. Network Layer
4. Transport Layer
5. Session Layer
6. Presentation Layer
7. Application Layer

OSI Model Characteristics

• An interface is present between any two layers.

• A layer may use the data present in another layer.

• Each layer is abstracted from other layers.

• The service provided by one layer can be used by the other layer.

• Two layers can provide same service e.g. Check Sum calculated at different layers.

• On two machines, six layers are logically connected except the physical layer. The physical layers of two machines are physically connected.

Internet and Packet Switching

Internet works on the concept of packet switching. Application layer passes data to the lower layer and that lower layer passes data to the next lower layer and on so on. In this data passing process through different layers, different headers are attached with the data which shows the source and destination addresses, number of data bytes in packet, type of

message etc. At physical layer, this packet is transmitted into the network. At reception, reverse procedure is adopted.

Fragmentation

When a packet is lost in the network, it is re-transmitted. If the size of the packet is large then retransmission of packet is wastage of resources and it also increases the delay in the network. To minimize this delay, a large packet is divided into small fragments. Each fragment contains a separate header having destination address and fragment number. This fragmentation effectively reduces the queuing delay. At destination, these fragments are re-assembled and data is sent to the application layer.

Routing

Routing works on store-and-forward policy. There are three methods used for routing:

- Source-based routing
- Virtual Circuit
- Destination-based routing

TCP/IP

Internet uses TCP/IP protocol. In the TCP/IP model, session and presentation layers are not present, s o Store-Forward routing is used.

WISH YOU BEST OF LUCK

KINDLY REMEMBER MUHAMMAD SAEED IN YOUR PRAYERS.