UNIT – 5

Memory Organization: Memory Hierarchy, Main Memory –RAM And ROM Chips, Memory Address map, Auxiliary memory-magnetic Disks, Magnetic tapes, Associate Memory,-Hardware Organization, Match Logic, Cache Memory –Associative Mapping , Direct Mapping, Set associative mapping ,Writing in to cache and cache Initialization , Cache Coherence ,Virtual memory-Address Space and memory Space ,Address mapping using pages, Associative memory page table ,page Replacement .

Memory Hierarchy

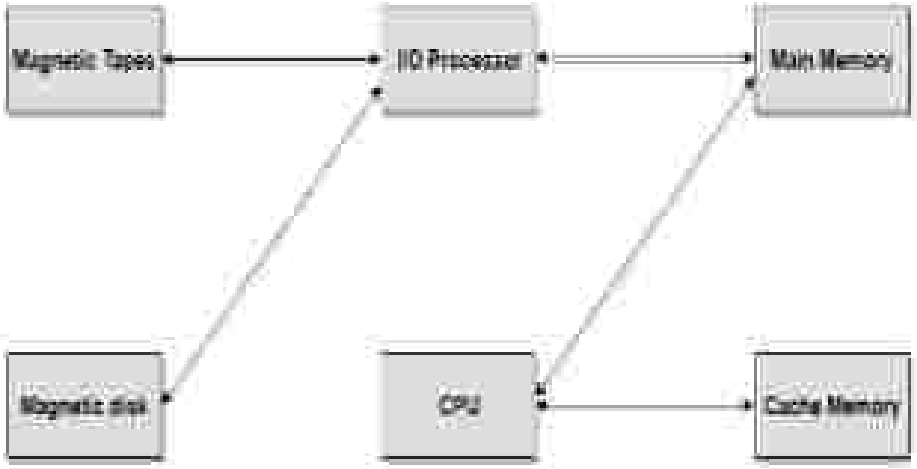
The total memory capacity of a computer can be visualized by hierarchy of components. The memory hierarchy system consists of all storage devices contained in a computer system from the slow Auxiliary Memory to fast Main Memory and to smaller Cache memory.

Auxiliary memory access time is generally **1000 times** that of the main memory, hence it is at the bottom of the hierarchy.

The **main memory** occupies the central position because it is equipped to communicate directly with the CPU and with auxiliary memory devices through Input/output processor (I/O).

When the program not residing in main memory is needed by the CPU, they are brought in from auxiliary memory. Programs not currently needed in main memory are transferred into auxiliary memory to provide space in main memory for other programs that are currently in use.

The **cache memory** is used to store program data which is currently being executed in the CPU. Approximate access time ratio between cache memory and main memory is about **1 to 7~10**



Memory Access Methods

Each memory type, is a collection of numerous memory locations. To access data from any memory, first it must be located and then the data is read from the memory location. Following are the methods to access information from memory locations:

1. **Random Access**: Main memories are random access memories, in which each memory location has a unique address. Using this unique address any memory location can be reached in the same amount of time in any order.
2. **Sequential Access**: This methods allows memory access in a sequence or in order.
3. **Direct Access**: In this mode, information is stored in tracks, with each track having a separate read/write head.

Main Memory

The memory unit that communicates directly within the CPU, Auxiliary memory and Cache memory, is called main memory. It is the central storage unit of the computer system. It is a large and fast memory used to store data during computer operations. Main memory is made up of **RAM** and **ROM**, with RAM integrated circuit chips holing the major share.

- RAM: Random Access Memory
  - **DRAM**: Dynamic RAM, is made of capacitors and transistors, and must be refreshed every 10~100 ms. It is slower and cheaper than SRAM.
  - **SRAM**: Static RAM, has a six transistor circuit in each cell and retains data, until powered off.

- **NVRAM**: Non-Volatile RAM, retains its data, even when turned off. Example: Flash memory.

- ROM: Read Only Memory, is non-volatile and is more like a permanent storage for information. It also stores the **bootstrap loader** program, to load and start the operating system when computer is turned on. **PROM**(Programmable ROM), **EPROM**(Erasable PROM) and **EEPROM**(Electrically Erasable PROM) are some commonly used ROMs.

Memory Address map:

- The addressing of memory can establish by means of a table that specifies the memory address assigned to each chip.
- The table, called a **memory address map**, is a pictorial representation of assigned address space for each chip in the system, shown in the table.
- To demonstrate with a particular example, assume that a computer system needs 512 bytes of RAM and 512 bytes of ROM.
  - The RAM and ROM chips to be used specified in figures.

Component	Hexa address	Address bus									
		10	9	8	7	6	5	4	3	2	1
RAM 1	0000 - 007F	0	0	0	x	x	x	x	x	x	x
RAM 2	0080 - 00FF	0	0	1	x	x	x	x	x	x	x
RAM 3	0100 - 017F	0	1	0	x	x	x	x	x	x	x
RAM 4	0180 - 01FF	0	1	1	x	x	x	x	x	x	x
ROM	0200 - 03FF	1	x	x	x	x	x	x	x	x	x

- The component column specifies whether a RAM or a ROM chip used.
- Moreover, The hexadecimal address column assigns a range of hexadecimal equivalent addresses for each chip.
- The address bus lines listed in the third column.
- Although there 16 lines in the address bus, the table shows only 10 lines because the other 6 not used in this example and assumed to be zero.
- The small x's under the address bus lines designate those lines that must connect to the address inputs in each chip.
- Moreover, The RAM chips have 128 bytes and need seven address lines. The ROM chip has 512 bytes and needs 9 address lines.
- The x's always assigned to the low-order bus lines: lines 1 through 7 for the RAM. And lines 1 through 9 for the ROM.
- It is now necessary to distinguish between four RAM chips by assigning to each a different address. For this particular example, we choose bus lines 8 and 9 to represent four distinct binary combinations.
- Also, The table clearly shows that the nine low-order bus lines constitute a memory space for RAM equal to $2^9 = 512$ bytes.
- The distinction between a RAM and ROM address done with another bus line. Here we choose line 10 for this purpose.
- When line 10 0, the CPU selects a RAM, and when this line equal to 1, it selects the ROM.

Auxiliary Memory

Devices that provide backup storage are called auxiliary memory. **For example:** Magnetic disks and tapes are commonly used auxiliary devices. Other devices used as auxiliary memory are magnetic drums, magnetic bubble memory and optical disks.

It is not directly accessible to the CPU, and is accessed using the Input/Output channels.

Cache Memory

The data or contents of the main memory that are used again and again by CPU, are stored in the cache memory so that we can easily access that data in shorter time.

Whenever the CPU needs to access memory, it first checks the cache memory. If the data is not found in cache memory then the CPU moves onto the main memory. It also transfers block of recent data into the cache and keeps on deleting the old data in cache to accomodate the new one.

Hit Ratio

The performance of cache memory is measured in terms of a quantity called **hit ratio**. When the CPU refers to memory and finds the word in cache it is said to produce a **hit**. If the word is not found in cache, it is in main memory then it counts as a **miss**.

The ratio of the number of hits to the total CPU references to memory is called hit ratio.

Hit Ratio = Hit/(Hit + Miss)

Associative Memory

It is also known as **content addressable memory (CAM)**. It is a memory chip in which each bit position can be compared. In this the content is compared in each bit cell which allows very fast table lookup. Since the entire chip can be compared, contents are randomly stored without considering addressing scheme. These chips have less storage capacity than regular memory chips.

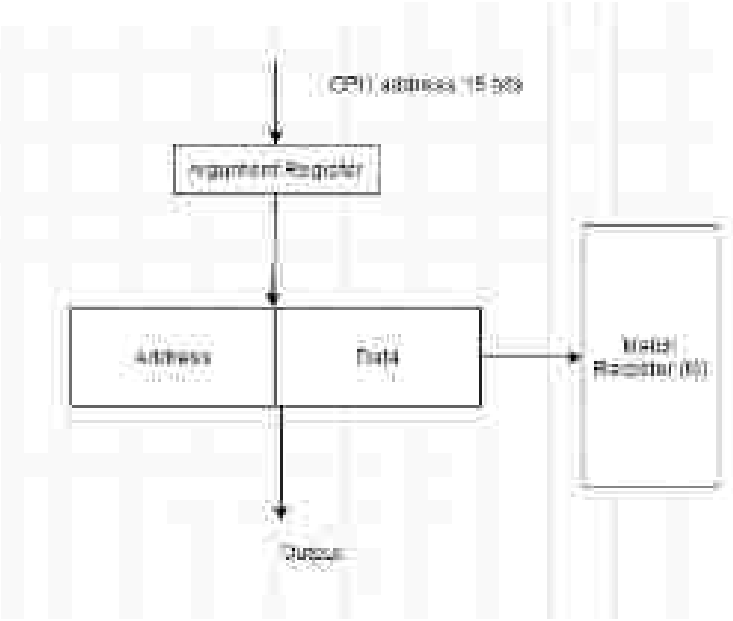
Memory Mapping and Concept of Virtual Memory

The transformation of data from main memory to cache memory is called mapping. There are 3 main types of mapping:

- Associative Mapping
- Direct Mapping
- Set Associative Mapping

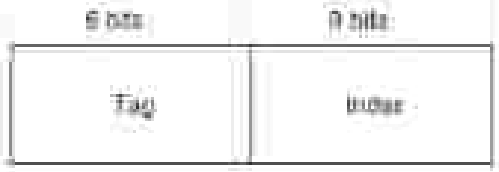
Associative Mapping

The associative memory stores both address and data. The address value of 15 bits is 5 digit octal numbers and data is of 12 bits word in 4 digit octal number. A CPU address of 15 bits is placed in **argument register** and the associative memory is searched for matching address.



Direct Mapping

The CPU address of 15 bits is divided into 2 fields. In this the 9 least significant bits constitute the **index** field and the remaining 6 bits constitute the **tag** field. The number of bits in index field is equal to the number of address bits required to access cache memory.



Set Associative Mapping

The disadvantage of direct mapping is that two words with same index address can't reside in cache memory at the same time. This problem can be overcome by set associative mapping.

In this we can store two or more words of memory under the same index address. Each data word is stored together with its tag and this forms a set.

Replacement Algorithms

Data is continuously replaced with new data in the cache memory using replacement algorithms. Following are the 2 replacement algorithms used:

- FIFO - First in First out. Oldest item is replaced with the latest item.
- LRU - Least Recently Used. Item which is least recently used by CPU is removed.

Writing in to cache and cache Initialization:

The benefit of write-through to main memory is that it simplifies the design of the computer system. With write-through, the main memory always has an up-to-date copy of the line. So when a read is done, main memory can always reply with the requested data.

If write-back is used, sometimes the up-to-date data is in a processor cache, and sometimes it is in main memory. If the data is in a processor cache, then that processor must stop main memory from replying to the read request, because the main memory might have a stale copy of the data. This is more complicated than write-through.

Also, write-through can simplify the cache coherency protocol because it doesn't need the *Modified* state. The *Modify* state records that the cache must write back the cache line before it invalidates or evicts the line. In write-through a cache line can always be invalidated without writing back since memory already has an up-to-date copy of the line.

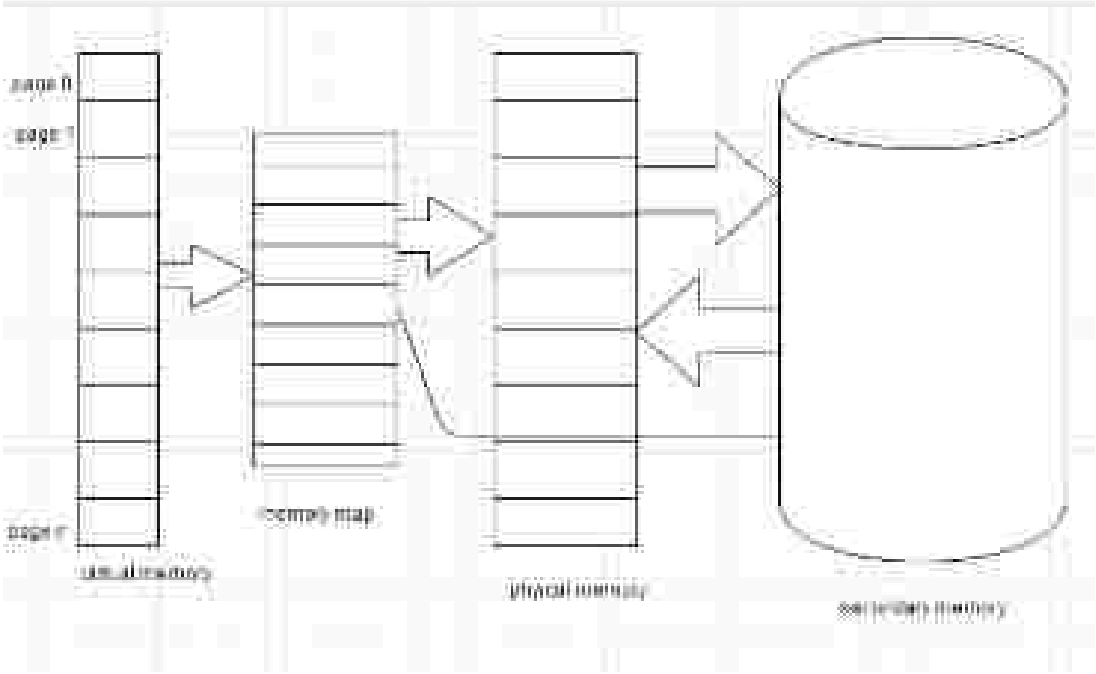
Cache Coherence:

In a shared memory multiprocessor with a separate cache memory for each processor , it is possible to have many copies of any one instruction operand : one copy in the main memory and one in each cache memory. When one copy of an operand is changed, the other copies of the operand must be changed also. Cache coherence is the discipline that ensures that changes in the values of shared operands are propagated throughout the system in a timely fashion.

Virtual Memory

Virtual memory is the separation of logical memory from physical memory. This separation provides large virtual memory for programmers when only small physical memory is available.

Virtual memory is used to give programmers the illusion that they have a very large memory even though the computer has a small main memory. It makes the task of programming easier because the programmer no longer needs to worry about the amount of physical memory available.



Address mapping using pages:

- The table implementation of the address mapping is simplified if the information in the address space and the memory space is each divided into groups of fixed size.
- Moreover, The physical memory is broken down into groups of equal size called blocks, which may range from 64 to 4096 words each.
- The term page refers to groups of address space of the same size.
- Also, Consider a computer with an address space of 8K and a memory space of 4K.
- If we split each into groups of 1K words we obtain eight pages and four blocks as shown in the figure.
- At any given time, up to four pages of address space may reside in main memory in any one of the four blocks.

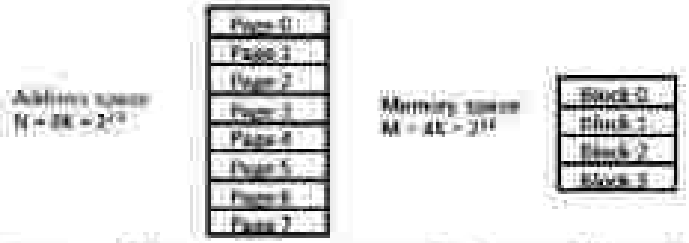
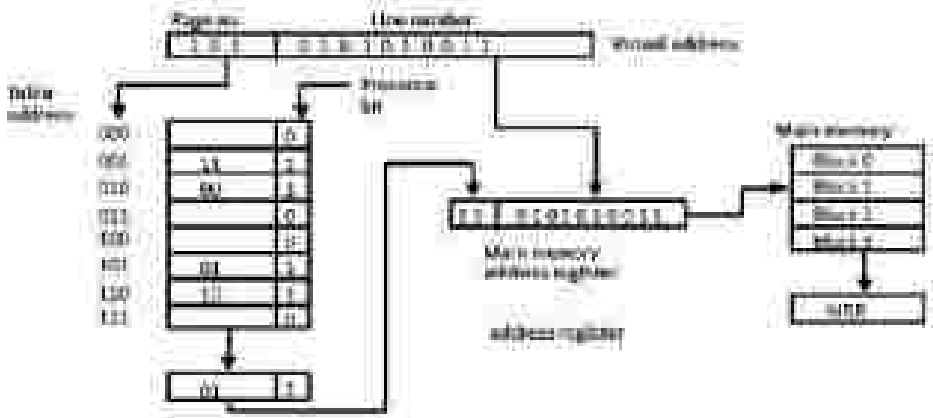


Figure Address and Memory space split into group of 1K words



Associative memory page table:

The implementation of the page table is vital to the efficiency of the virtual memory technique, for each memory reference must also include a reference to the page table. The fastest solution is a set of dedicated registers to hold the page table but this method is impractical for large page tables because of the expense. But keeping the page table in main memory could cause intolerable delays because even only one memory access for the page table involves a slowdown of 100 percent and large page tables can require more than one memory access. The solution is to augment the page table with special high-speed memory made up of associative registers or translation look aside buffers (TLBs) which are called ASSOCIATIVE MEMORY.

Page replacement

The advantage of virtual memory is that processes can be using more memory than exists in the machine; when memory is accessed that is not present (a **page fault**), it must be paged in (sometimes referred to as being "swapped in", although some people reserve "swapped in to refer to bringing in an entire address space).

Swapping in pages is very expensive (it requires using the disk), so we'd like to avoid page faults as much as possible. The algorithm that we use to choose which pages to evict to make space for the new page can have a large impact on the number of page faults that occur.

UNIT – 5

Input-Output Organization: Peripheral Devices, Input-Output Interface, Asynchronous data transfer Modes of Transfer, Priority Interrupt Direct memory Access, Input –Output Processor (IOP)
Pipeline And Vector Processing: Parallel Processing, Pipelining, Arithmetic Pipeline, Instruction Pipeline, Dependencies, Vector Processing.

Introduction:

The I/O subsystem of a computer provides an efficient mode of communication between the central system and the outside environment. It handles all the input-output operations of the computer system.

Peripheral Devices

Input or output devices that are connected to computer are called **peripheral devices**. These devices are designed to read information into or out of the memory unit upon command from the CPU and are considered to be the part of computer system. These devices are also called **peripherals**.

For example: *Keyboards*, *display units* and *printers* are common peripheral devices.

There are three types of peripherals:

1. **Input peripherals** : Allows user input, from the outside world to the computer. Example: Keyboard, Mouse etc.
2. **Output peripherals**: Allows information output, from the computer to the outside world. Example: Printer, Monitor etc
3. **Input-Output peripherals**: Allows both input(from outised world to computer) as well as, output(from computer to the outside world). Example: Touch screen etc.

Interfaces

Interface is a shared boundary btween two separate components of the computer system which can be used to attach two or more components to the system for communication purposes.

There are two types of interface:

1. CPU Inteface
2. I/O Interface

Let's understand the I/O Interface in details,

Input-Output Interface

Peripherals connected to a computer need special communication links for interfacing with CPU. In computer system, there are special hardware components between the CPU and peripherals to control or manage the input-output transfers. These components are called **input-output interface units** because they provide communication links between processor bus and peripherals. They provide a method for transferring information between internal system and input-output devices.

Asynchronous Data Transfer

We know that, the internal operations in individual unit of digital system are synchronized by means of clock pulse, means clock pulse is given to all registers within a unit, and all data transfer among internal registers occur simultaneously during occurrence of clock pulse.Now, suppose any two units of digital system are designed independently such as CPU and I/O interface.

And if the registers in the interface(I/O interface) share a common clock with CPU registers, then transfer between the two units is said to be synchronous.But in most cases, the internal timing in each unit is independent from each other in such a way that each uses its own private clock for its internal registers.In that case, the two units are said to be asynchronous to each other, and if data transfer occur between them this data transfer is said to be **Asynchronous Data Transfer**.

But, the Asynchronous Data Transfer between two independent units requires that control signals be transmitted between the communicating units so that the time can be indicated at which they send data.

This asynchronous way of data transfer can be achieved by two methods:

1. One way is by means of strobe pulse which is supplied by one of the units to other unit.When transfer has to occur.This method is known as "**Strobe Control**".
2. Another method commonly used is to accompany each data item being transferred with a control signal that indicates the presence of data in the bus.The unit receiving the data item responds with another signal to acknowledge receipt of the data.This method of data transfer between two independent units is said to be "**Handshaking**".

The strobe pulse and handshaking method of asynchronous data transfer are not restricted to I/O transfer.In fact, they are used extensively on numerous occasion requiring transfer of data between two independent units.So, here we consider the transmitting unit as source and receiving unit as destination.
As an example: The CPU, is the source during an output or write transfer and is the destination unit during input or read transfer.

And thus, the sequence of control during an asynchronous transfer depends on whether the transfer is initiated by the source or by the destination.

So, while discussing each way of data transfer asynchronously we see the sequence of control in both terms when it is initiated by source or when it is initiated by destination.In this way, each way of data transfer, can be further divided into parts, source initiated and destination initiated.

We can also specify, asynchronous transfer between two independent units by means of a timing diagram that shows the timing relationship that exists between the control and the data buses.

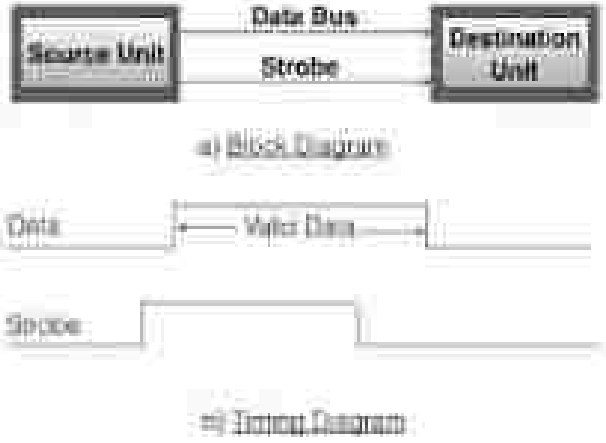
Now, we will discuss each method of asynchronous data transfer in detail one by one.

1. Strobe Control:

The Strobe Control method of asynchronous data transfer employs a single control line to time each transfer. This control line is also known as strobe and it may be achieved either by source or destination, depending on which initiate transfer.

Source initiated strobe for data transfer:

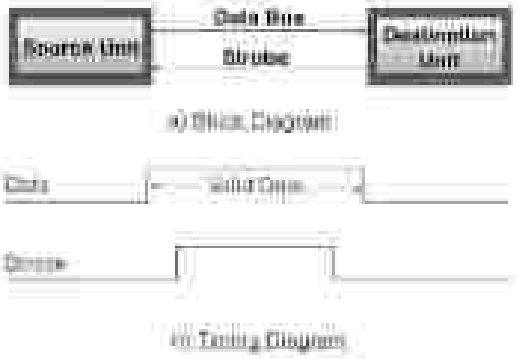
The block diagram and timing diagram of strobe initiated by source unit is shown in figure below:



In block diagram we see that strobe is initiated by source, and as shown in timing diagram, the source unit first places the data on the data bus.After a brief delay to ensure that the data settle to a steady value, the source activates a strobe pulse.The information on data bus and strobe control signal remain in the active state for a sufficient period of time to allow the destination unit to receive the data.Actually, the destination unit, uses a falling edge of strobe control to transfer the contents of data bus to one of its internal registers.The source removes the data from the data bus after it disables its strobe pulse.New valid data will be available only after the strobe is enabled again.

Destination-initiated strobe for data transfer:

The block diagram and timing diagram of strobe initiated by destination is shown in figure below:



In block diagram, we see that, the strobe initiated by destination, and as shown in timing diagram, the destination unit first activates the strobe pulse, informing the source to provide the data.The source unit responds by placing the requested binary information on the data bus.The data must be valid and remain in the bus long enough for the destination unit to accept it.The falling edge of strobe pulse can be used again to trigger a destination register.The destination unit then disables the strobe.And source removes the data from data bus after a per determine time interval.

Now, actually in computer, in the first case means in strobe initiated by source – the strobe may be a memory-write control signal from the CPU to a memory unit.The source, CPU, places the word on the data bus and informs the memory unit, which is the destination, that this is a write operation.

And in the second case i.e, in the strobe initiated by destination – the strobe may be a memory read control from the CPU to a memory unit.The destination, the CPU, initiates the read operation to inform the memory, which is a source unit, to place selected word into the data bus.

2. Handshaking:

The disadvantage of strobe method is that source unit that initiates the transfer has no way of knowing whether the destination has actually received the data that was placed in the bus.Similarly, a destination unit that initiates the transfer has no way of knowing whether the source unit, has actually placed data on the bus.

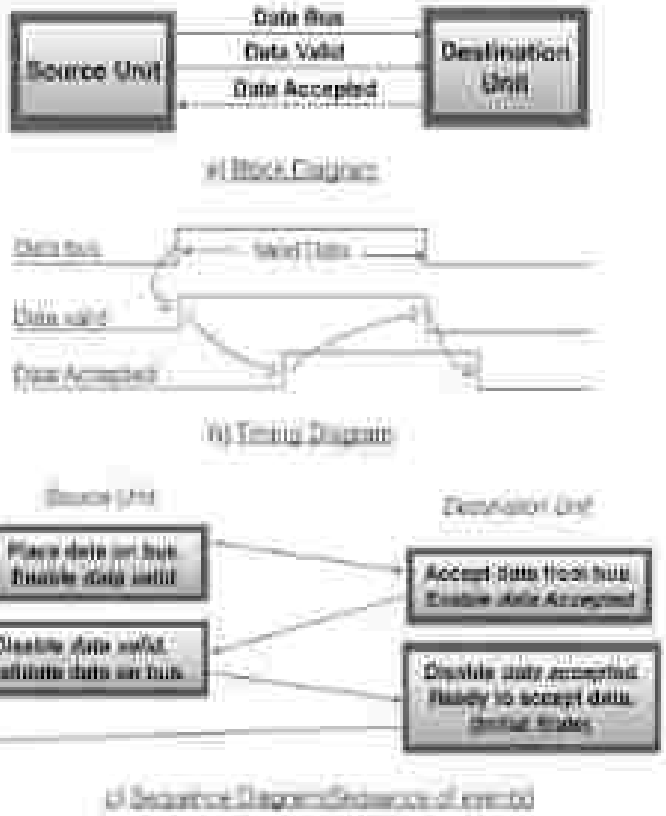
This problem can be solved by handshaking method.

Hand shaking method introduce a second control signal line that provides a replay to the unit that initiates the transfer.

In it, one control line is in the same direction as the data flow in the bus from the source to destination.It is used by source unit to inform the destination unit whether there are valid data in the bus.The other control line is in the other direction from destination to the source.It is used by the destination unit to inform the source whether it can accept data.And in it also, sequence of control depends on unit that initiate transfer.Means sequence of control depends whether transfer is initiated by source and destination.Sequence of control in both of them are described below:

Source initiated Handshaking:

The source initiated transfer using handshaking lines is shown in figure below:



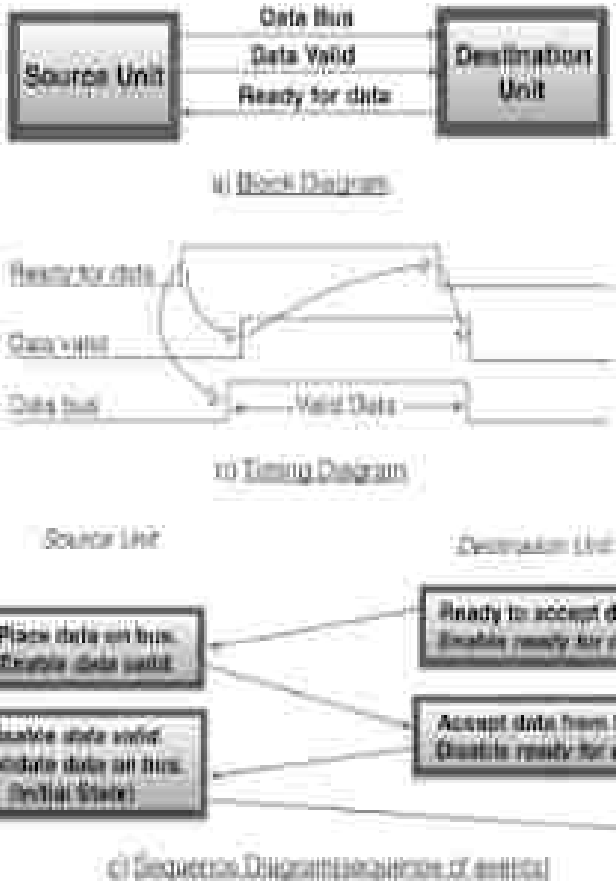
In its block diagram, we se that two handshaking lines are "data valid", which is generated by the source unit, and "data accepted", generated by the destination unit.

The timing diagram shows the timing relationship of exchange of signals between the two units.Means as shown in its timing diagram, the source initiates a transfer by placing data on the bus and enabling its data valid signal.The data accepted signal is then activated by destination unit after it accepts the data from the bus.The source unit then disable its data valid signal which invalidates the data on the bus.After this, the destination unit disables its data accepted signal and the system goes into initial state.The source unit does not send the next data item until after the destination unit shows its readiness to accept new data by disabling the data accepted signal.

This sequence of events described in its sequence diagram, which shows the above sequence in which the system is present, at any given time.

Destination initiated handshaking:

The destination initiated transfer using handshaking lines is shown in figure below:



In its block diagram, we see that the two handshaking lines are "data valid", generated by the source unit, and "ready for data" generated by destination unit.Note that the name of signal data accepted generated by destination unit has been changed to ready for data to reflect its new meaning.

In it, transfer is initiated by destination, so source unit does not place data on data bus until it receives ready for data signal from destination unit.After that, hand shaking process is some as that of source initiated.

The sequence of event in it are shown in its sequence diagram and timing relationship between signals is shown in its timing diagram.

Thus, here we can say that, sequence of events in both cases would be identical.If we consider ready for data signal as the complement of data accept.Means, the only difference between source and destination initiated transfer is in their choice of initial state.

Modes of I/O Data Transfer

Data transfer between the central unit and I/O devices can be handled in generally three types of modes which are given below.

1. Programmed I/O
2. Interrupt Initiated I/O
3. Direct Memory Access

Programmed I/O

Programmed I/O instructions are the result of I/O instructions written in computer program. Each data item transfer is initiated by the instruction in the program.

Usually the program controls data transfer to and from CPU and peripheral. Transferring data under programmed I/O requires constant monitoring of the peripherals by the CPU.

Interrupt Initiated I/O

In the programmed I/O method the CPU stays in the program loop until the I/O unit indicates that it is ready for data transfer. This is time consuming process because it keeps the processor busy needlessly.

This problem can be overcome by using **interrupt initiated I/O**. In this when the interface determines that the peripheral is ready for data transfer, it generates an interrupt. After receiving the interrupt signal, the CPU stops the task which it is processing and service the I/O transfer and then returns back to its previous processing task.

Direct Memory Access

Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This technique is known as **DMA**.

In this, the interface transfer data to and from the memory through memory bus. A DMA controller manages to transfer data between peripherals and memory unit.

Many hardware systems use DMA such as disk drive controllers, graphic cards, network cards and sound cards etc. It is also used for intra chip data transfer in multicore processors. In DMA, CPU would initiate the transfer, do other operations while the transfer is in progress and receive an interrupt from the DMA controller when the transfer has been completed.

Priority Interrupt

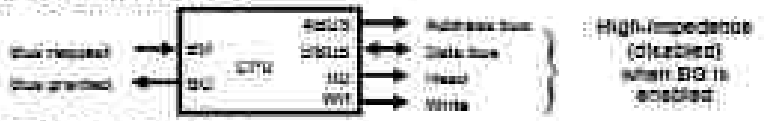
A priority interrupt is a system which decides the priority at which various devices, which generates the interrupt signal at the same time, will be serviced by the CPU. The system has authority to decide which conditions are allowed to interrupt the CPU, while some other interrupt is being serviced. Generally, devices with high speed transfer such as *magnetic disks* are given high priority and slow devices such as *keyboards* are given low priority.

When two or more devices interrupt the computer simultaneously, the computer services the device with the higher priority first.

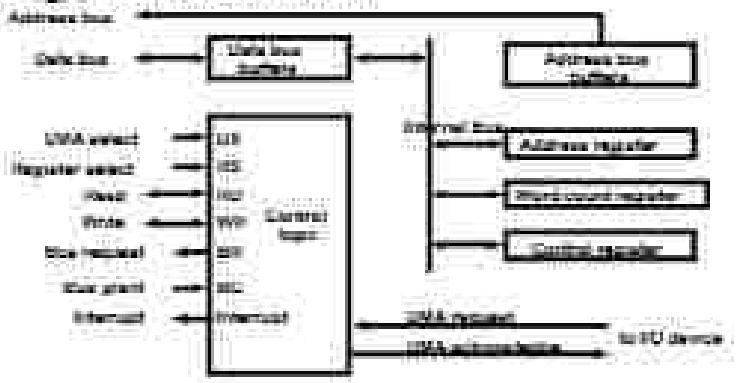
DIRECT MEMORY ACCESS

Block of data transfer from high speed devices, Drum, Disk, Tape

CPU bus signals for DMA transfer

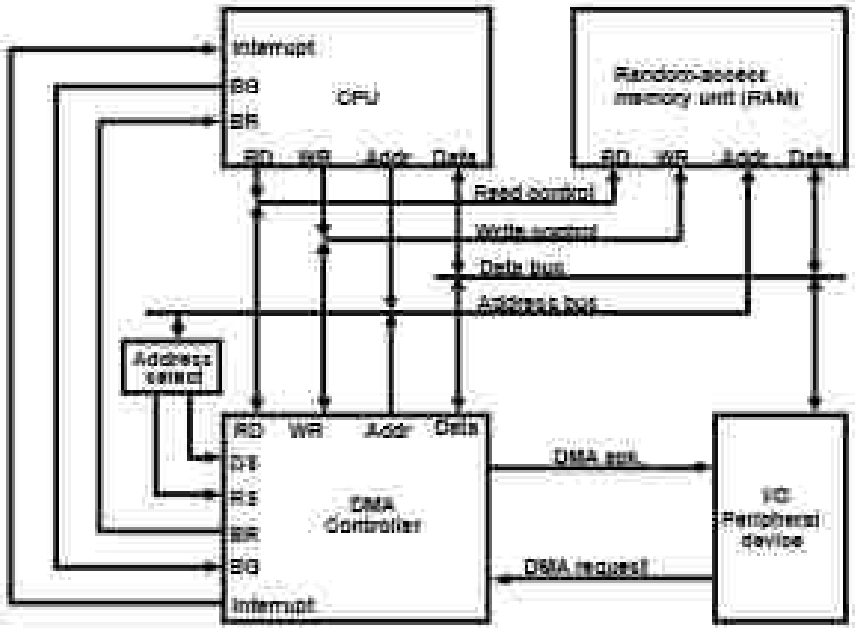


Block diagram of DMA controller



* DMA controller - Interface which allows I/O transfer directly between Memory and Device, freeing CPU for other tasks
* CPU initializes DMA Controller by sending memory address and the block size(number of words)

DMA TRANSFER



Input/output Processor

An input-output processor (IOP) is a processor with direct memory access capability. In this, the computer system is divided into a memory unit and number of processors.

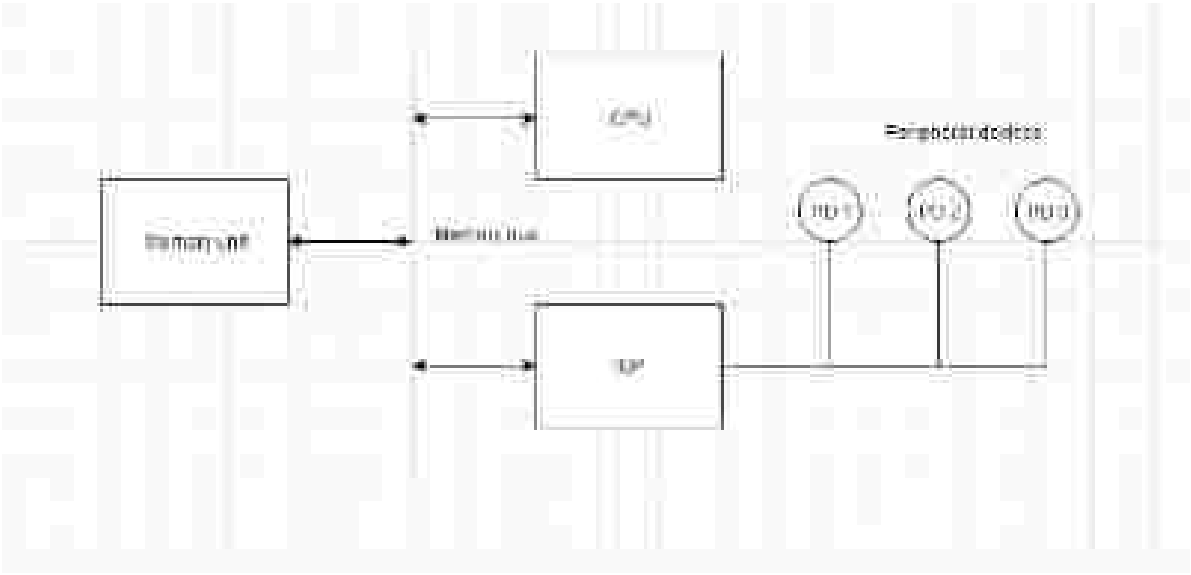
Each IOP controls and manage the input-output tasks. The IOP is similar to CPU except that it handles only the details of I/O processing. The IOP can fetch and execute its own instructions. These IOP instructions are designed to manage I/O transfers only.

Block Diagram Of I/O Processor:

Below is a block diagram of a computer along with various I/O Processors. The memory unit occupies the central position and can communicate with each processor.

The CPU processes the data required for solving the computational tasks. The IOP provides a path for transfer of data between peripherals and memory. The CPU assigns the task of initiating the I/O program.

The IOP operates independent from CPU and transfer data between peripherals and memory.



The communication between the IOP and the devices is similar to the program control method of transfer. And the communication with the memory is similar to the direct memory access method.

In large scale computers, each processor is independent of other processors and any processor can initiate the operation.

The CPU can act as master and the IOP act as slave processor. The CPU assigns the task of initiating operations but it is the IOP, who executes the instructions, and not the CPU. CPU instructions provide operations to start an I/O transfer. The IOP asks for CPU through interrupt.

Instructions that are read from memory by an IOP are also called *commands* to distinguish them from instructions that are read by CPU. Commands are prepared by programmers and are stored in memory. Command words make the program for IOP. CPU informs the IOP where to find the commands in memory.