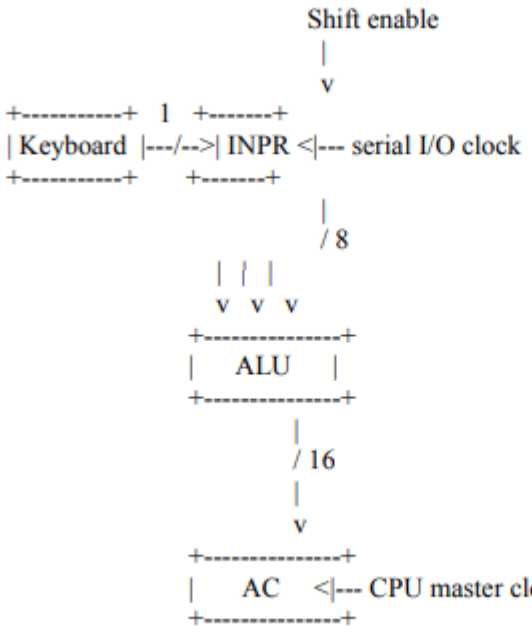
UNIT 4:

Input-Output Organization: Input-Output Interface, Asynchronous data transfer, Modes of Transfer, Priority Interrupt Direct memory Access.
Memory Organization: Memory Hierarchy, Main Memory, Auxiliary memory, Associate Memory, Cache Memory.

The Basic Computer I/O consists of a simple terminal with a keyboard and a printer/monitor. The keyboard is connected serially (1 data wire) to the INPR register. INPR is a shift register capable of shifting in external data from the keyboard one bit at a time. INPR outputs are connected in parallel to the ALU.



How many CPU clock cycles are needed to transfer a character from the keyboard to the INPR register? (tricky)

Are the clock pulses provided by the CPU master clock?

RS232, USB, Firewire are serial interfaces with their own clock independent of the CPU. (USB speed is independent of processor speed.)

□ RS232: 115,200 kbps (some faster)

□ USB: 11 mbps

□ USB2: 480 mbps

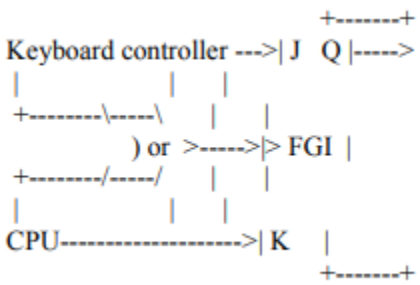
□ FW400: 400 mbps

□ FW800: 800 mbps

□ USB3: 4.8 gbps

OUTR inputs are connected to the bus in parallel, and the output is connected serially to the terminal. OUTR is another shift register, and the printer/monitor receives an end-bit during each clock pulse.

I/O Operations Since input and output devices are not under the full control of the CPU (I/O events are asynchronous), the CPU must somehow be told when an input device has new input ready to send, and an output device is ready to receive more output. The FGI flip-flop is set to 1 after a new character is shifted into INPR. This is done by the I/O interface, not by the control unit. This is an example of an asynchronous input event (not synchronized with or controlled by the CPU). The FGI flip-flop must be cleared after transferring the INPR to AC. This must be done as a microoperation controlled by the CU, so we must include it in the CU design. The FGO flip-flop is set to 1 by the I/O interface after the terminal has finished displaying the last character sent. It must be cleared by the CPU after transferring a character into OUTR. Since the keyboard controller only sets FGI and the CPU only clears it, a JK flip-flop is convenient:



How do we control the CK input on the FGI flip-flop? (Assume leading-edge triggering.) There are two common methods for detecting when I/O devices are ready, namely software polling and interrupts. These two methods are discussed in the following sections. Table 5-5 outlines the Basic Computer input-output instructions.

Interrupts To alleviate the problems of software polling, a hardware solution is needed. 108 Analogies to software polling in daily life tend to look rather silly. For example, imagine a teacher is analogous to a CPU, and the students are I/O devices. The students are working asynchronously, as the teacher walks around the room constantly asking each individual student "are you done yet?". What would be a better approach? With interrupts, the running program is not responsible for

checking the status of I/O devices. Instead, it simply does its own work, and assumes that I/O will take care of itself! When a device becomes ready, the CPU hardware initiates a branch to an I/O subprogram called an interrupt service routine (ISR), which handles the I/O transaction with the device. An interrupt can occur during any instruction cycle as long as interrupts are enabled. When the current instruction completes, the CPU interrupts the flow of the program, executes the ISR, and then resumes the program. The program itself is not involved and is in fact unaware that it has been interrupted. Figure 5-13 outlines the Basic Computer interrupt process. Interrupts can be globally enabled or disabled via the IEN flag (flip-flop). Some architectures have a separate ISR for each device. The Basic Computer has a single ISR that services both the input and output devices. If interrupts are enabled, then when either FGI or FGO gets set, the R flag also gets set. (R = FGI v FGO) This allows the system to easily check whether any I/O device needs service. Determining which one needs service can be done by the ISR. If R = 0, the CPU goes through a normal instruction cycle. If R = 1, the CPU branches to the ISR to process an I/O transaction. How much time does checking for interrupts add to the instruction cycle? Interrupts are usually disabled while the ISR is running, since it is difficult to make an ISR reentrant. (Callable while it is already in progress, such as a recursive function.) Hence, IEN and R are cleared as part of the interrupt cycle. IEN should be re-enabled by the ISR when it is finished. (In many architectures this is done by a special return instruction to ensure that interrupts are not enabled before the return is actually executed.) The Basic Computer interrupt cycle is shown in figure 5-13 (above). The Basic Computer interrupt cycle in detail: T0'T1'T2'(IEN)(FGI v FGO): R ← 1 109 RT0: AR ← 0, TR ← PC RT1: M[AR] ← TR, PC ← 0 RT2: PC ← PC + 1, IEN ← 0, R ← 0, SC ← 0 To enable the use of interrupts requires several steps: 1. Write an ISR 2. Install the ISR in memory at some arbitrary address X 3. Install the instruction "BUN X" at address 1 4. Enable interrupts with the ION instruction The sequence of events utilizing an interrupt to process keyboard input is as follows: 1. A character is typed 2. FGI ← 1 (same as with polling) 3. R ← 1, IEN ← 0 4. M[0] ← PC (store return address) 5. PC ← 1 (branch to interrupt vector) 6. BUN X (branch to ISR) 7. ISR checks FGI (found to be 1) 8. INP (AC ← INPR) 9. Character in AC is placed in a queue 10. ISR checks FGO (found to be 0) 11. ION 12. BUN 0 I Programs then read their input from a queue rather than directly from the input device. The ISR adds input to the queue as soon as it is typed, regardless of what code is running, and then returns to the running program.

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.

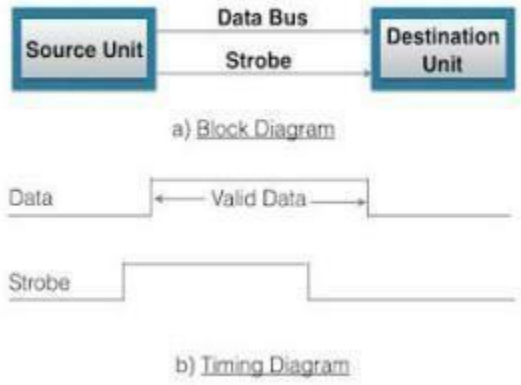
This asynchronous way of data transfer can be achieved by two methods: 1. One way is by means of strobe pulse which is supplied by one of the units to other unit.When transfer has to occur.This method is known as "Strobe Control". 2. Another method commonly used is to accompany each data item being transferred 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.

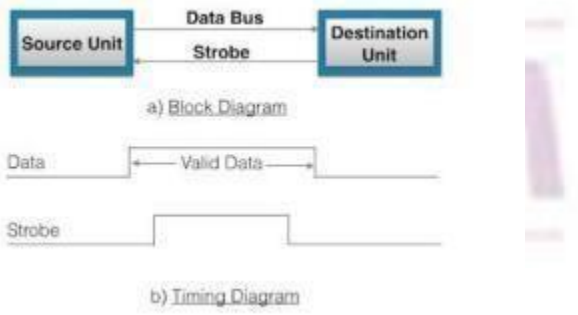
1. Strobe Control:

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:



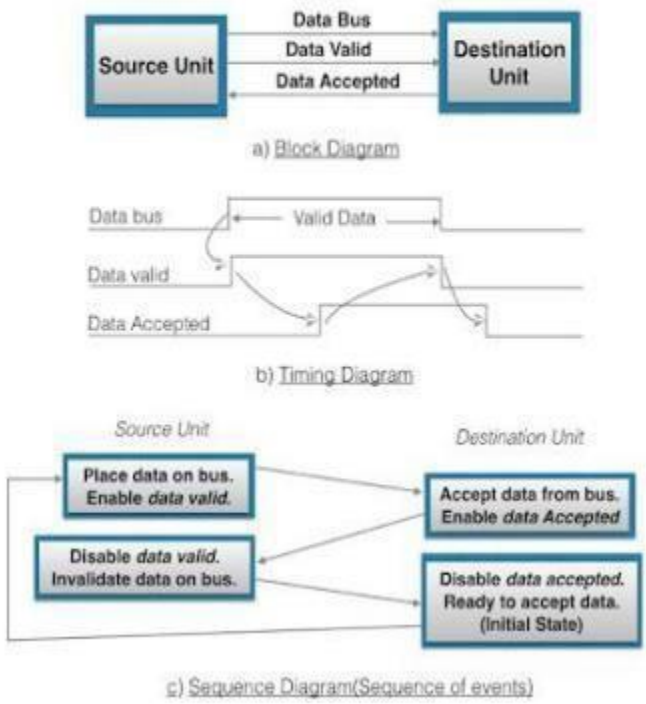
In block diagram, we see that, the strobe initiated by destination, and as shown in timing diagram, the destination unit first activates the strobe pulse, informing the source to provide the data.The source unit responds by placing the requested binary information on the data bus.The data must be valid 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.

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

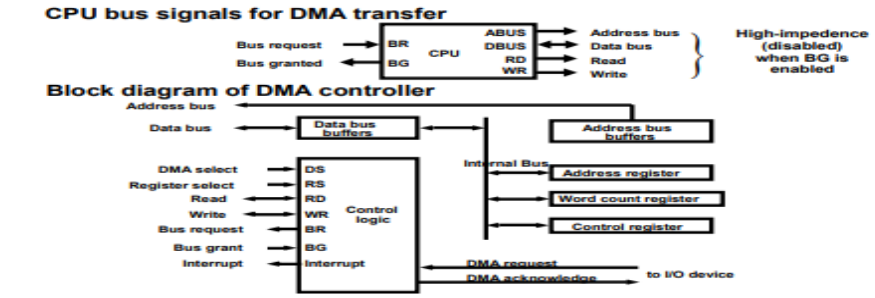
2. Interrupt Initiated I/O

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.

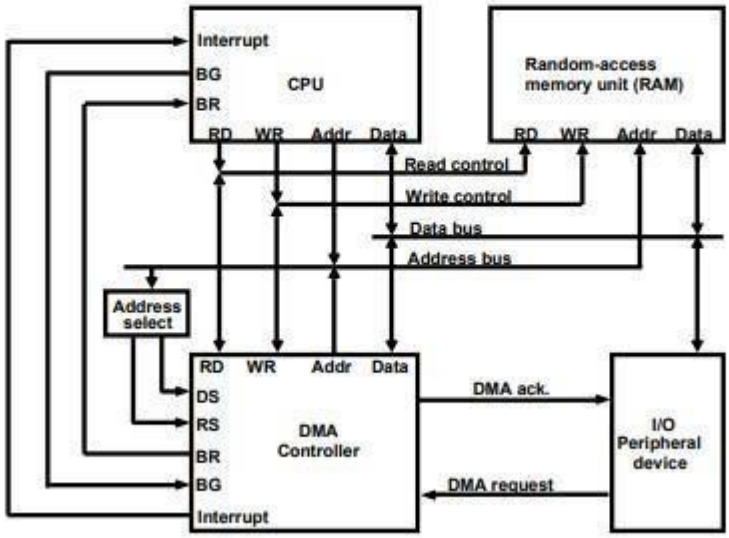
DIRECT MEMORY ACCESS

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



* 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



MEMORY ORGANIZATION

• RAM composed of a large number of (2M) of addressable locations, each of which stores a w-bit word.

• RAM operates as follows: first the address of the target location to be accessed is transferred via the address bus to the RAM's address buffer.

• The address is then processed by the address decoder, which selects the required location in the storage cell unit.

• If a read operation is requested, the contents of the addressed location are transferred from the storage cell unit to the data buffer and from there to the data bus.

• If a write operation is requested, the word to be stored is transferred from the data bus to the selected location in the stored unitThe storage unit is made up of many identical 1-bit memory cells and their Interconnections. In each line connected to the storage cell unit, we can expect to

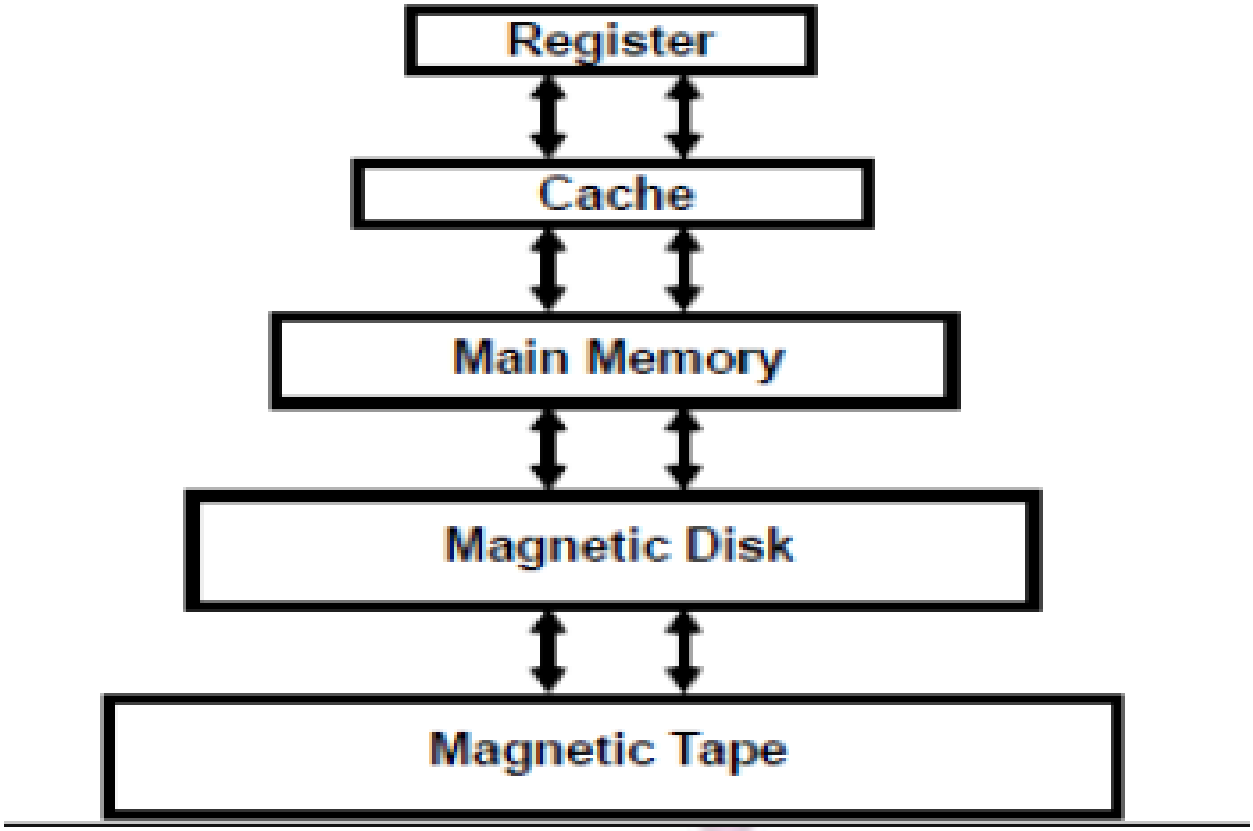
• find a driver that acts as either an amplifier or a transducer of physical signals. Organization

• assume that each word is stored in a single track and that each access results In the transfer of a block of words.

• The address of the data to be accessed is applied to the address decoder, whose output determines the track to be used and the location of the desired block of Information within the track.

• the track address determines the particular read-write head to be selected.The selected head is moved into position to transfer data to of from the target track. A track position indicator generates the address of the block that isCurrently passing the read-write head.

• The generated address is compared with the block address produced by the address decoder.The selected head is enabled and the data transfer between the storage track and the memory data buffer register begins.



• The read-write head is disabled when a complete block information has been transferred

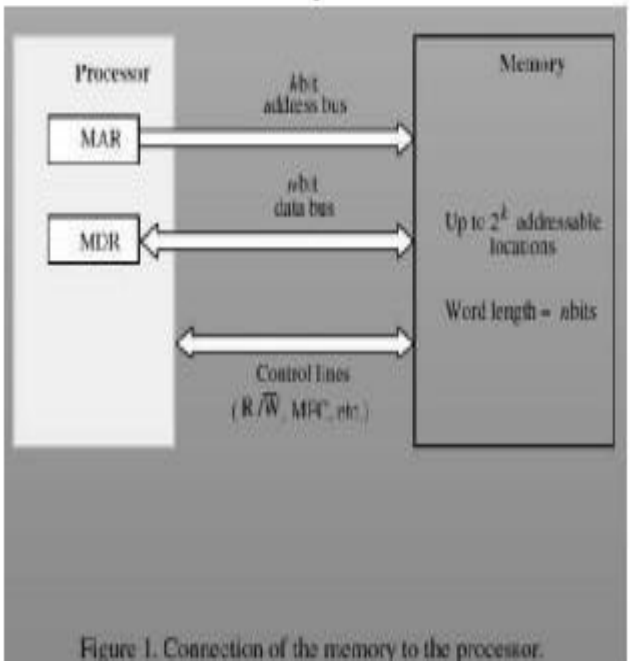


Figure 1. Connection of the memory to the processor.

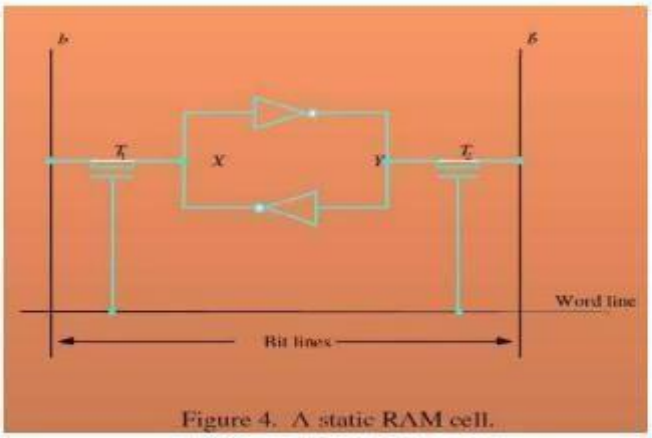


Figure 4. A static RAM cell.

Static memories (RAM)

Circuits capable of retaining their state as long as power is applied

Static RAM(SRAM)

volatile

DRAMS:

Charge on a capacitor

Needs —Refreshing

Synchronous DRAMs

Synchronized with a clock signal

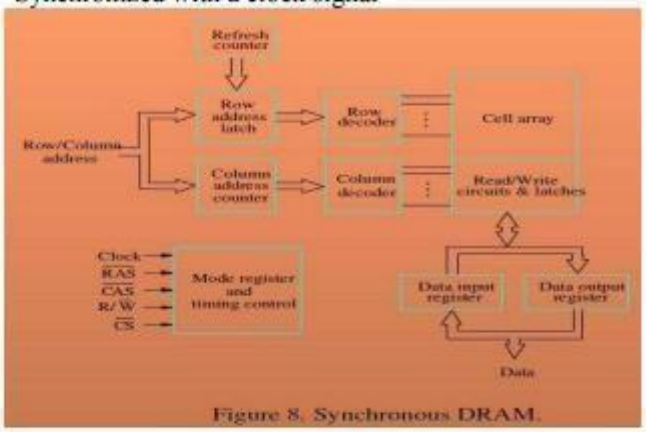


Figure 8. Synchronous DRAM.

Memory system considerations

– Cost
– Speed
– Power dissipation
– Size of chip

Principle of locality:

Temporal locality (locality in time): If an item is referenced, it will tend to be referenced again soon.

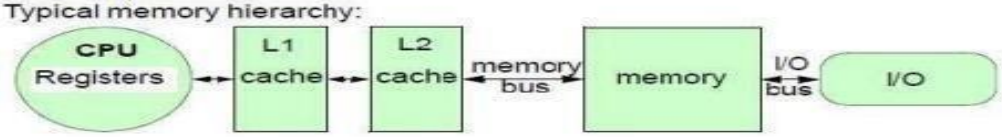
Spatial locality (locality in space): If an item is referenced, items whose addresses are close by will tend to be referenced soon.

Sequentiality (subset of spatial locality).

The principle of locality can be exploited implementing the memory of computer as a *memory hierarchy*, taking advantage of all types of memories.

Method: The level closer to processor (the fastest) is a subset of any level further away, and all the data is stored at the lowest level (the slowest).

Typical memory hierarchy:



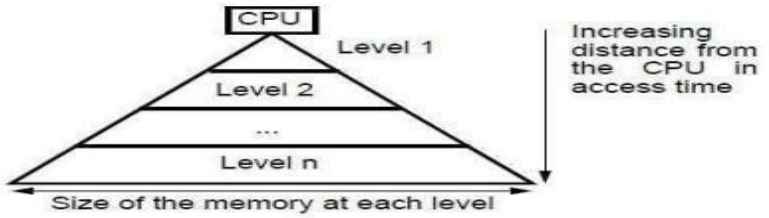
Level	1	2	3	4
Named as	Registers	Cache	memory	disk storage
Typical size	<1 KB	< 4 MB	<2 GB	>2GB
Access time (ns)	2 - 5	3 - 10	80 - 400	5'000'000
Bandwidth(MB/sec)	4000 - 32'000	800 - 5000	400 - 2000	4 - 32
Managed by	Compiler	Hardware	Operating system	Operating system / user

Cache Memories

– Speed of the main memory is very low in comparison with the speed of processor – For good performance, the processor cannot spend much time of its time waiting to access instructions and data in main memory. – Important to device a scheme that reduces the time to ace the information – An efficient solution is to use fast cache memory When a cache is full and a memory word 101 that is not in the cache is referenced, the cache control hardware must decide which block should be removed to create space for the new block that contain the referenced word.

The basics of Caches

" The caches are organized on basis of *blocks*, the smallest amount of data which can be copied between two adjacent levels at a time.
" If data requested by the processor is present in some block in the upper level,
it is called a *hit*.
" If data is not found in the upper level, the request is called a *miss* and the data is retrieved from the lower level in the hierarchy.
" The fraction of memory accesses found in the upper level is called a *hit ratio*.
" The storage, which takes advantage of locality of accesses is called a *cache*



Amdahl's Law about overall speedup:

$$\text{Speedup} = \frac{1}{(1 - \text{fraction of time cache can be used}) + \frac{\text{fraction of time cache can be used}}{\text{Speedup using cache}}}$$

Alternatively, CPU stalls can be considered[1]:

CPU execution time = (CPU clock cycles + Memory stall cycles) × Clock cycle

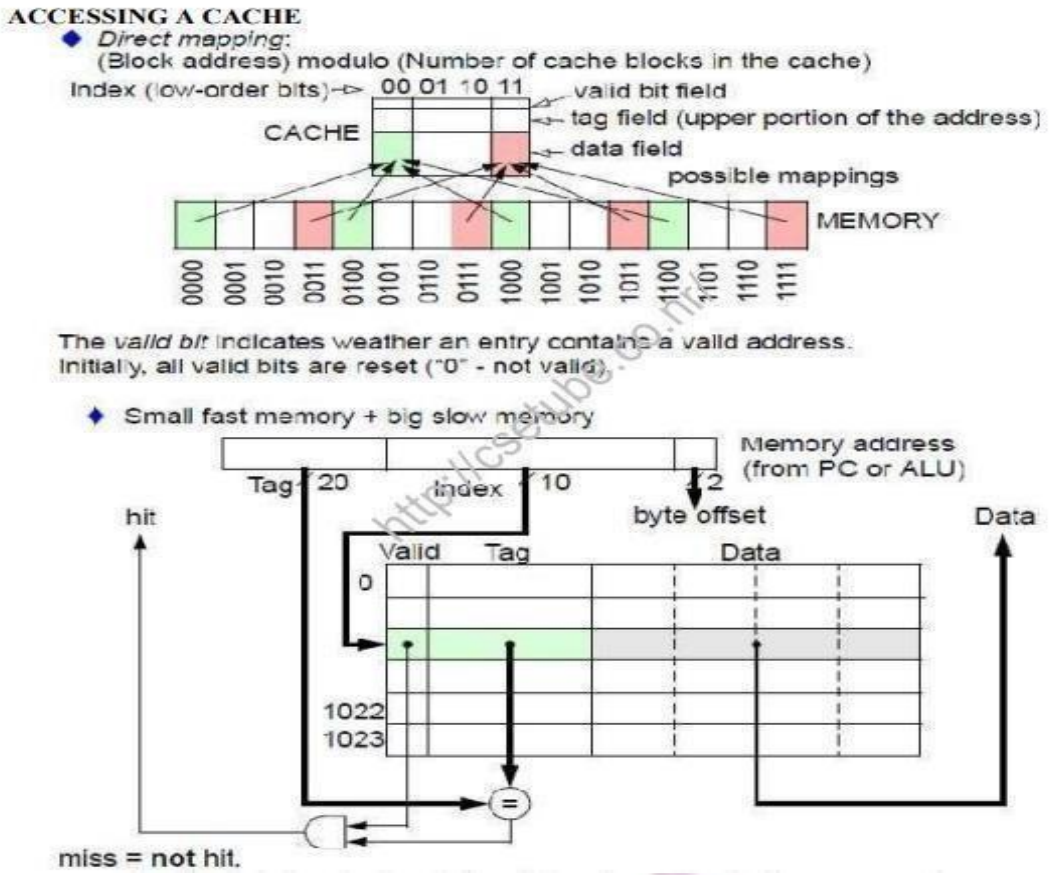
The number of memory stall cycles depends:

Memory stall cycles = IC × Memory references per instruction × Miss rate × Miss penalty

Size	Instruction cache	Data cache	
1 KB	3.06%	24.61%	13.34%
4 KB	1.76%	15.94%	7.24%
16 KB	0.64%	6.47%	2.87%
64 KB	0.15%	3.77%	1.35%

Table: Direct mapped cache, 32-byte blocks, SPEC92, DECstation 5000.

1. Here is assumed that CPU clock cycles include the time to handle a cache hit, and the CPU is stalled during a cache miss.



Virtual memory It a computer system technique which gives an application program the impression that it has contiguous working memory (an address space), while in fact it may be physically fragmented and may even overflow on to disk storage. Virtual memory provides two primary functions: 1. Each process has its own address space, thereby not required to be relocated nor required to use relative addressing mode. 2. Each process sees one contiguous block of free memory upon launch. Fragmentation is hidden.

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

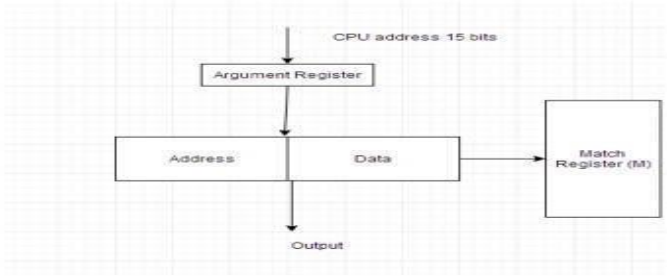
The data or contents of the main memory that are used again and again by CPU, are stored in

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.

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:

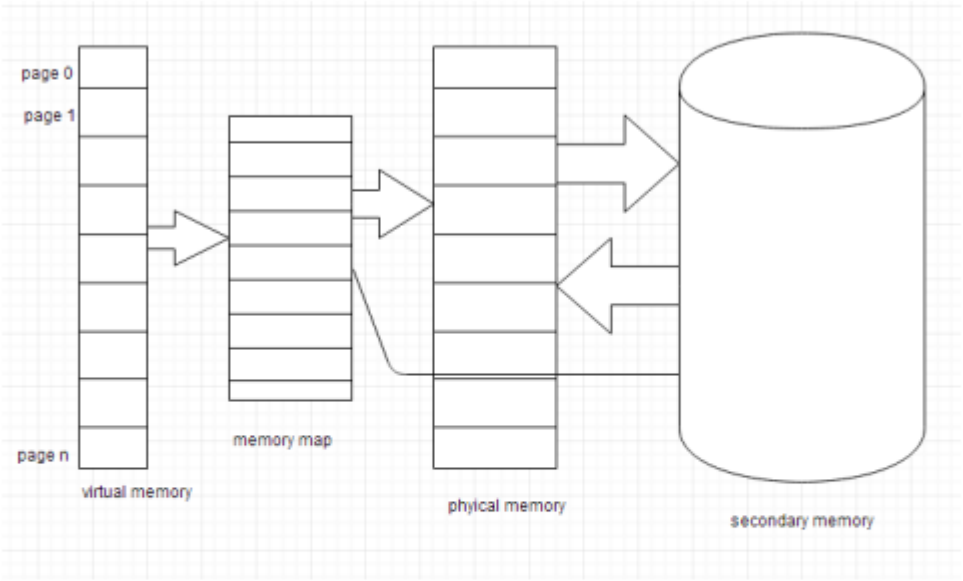
The benefit of write-through to main memory is that it simplifies the design of the computer system. With write-through, the main memory always has an up-to-date copy of the line. So when a read is done, main memory can always reply with the requested data.

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.

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.