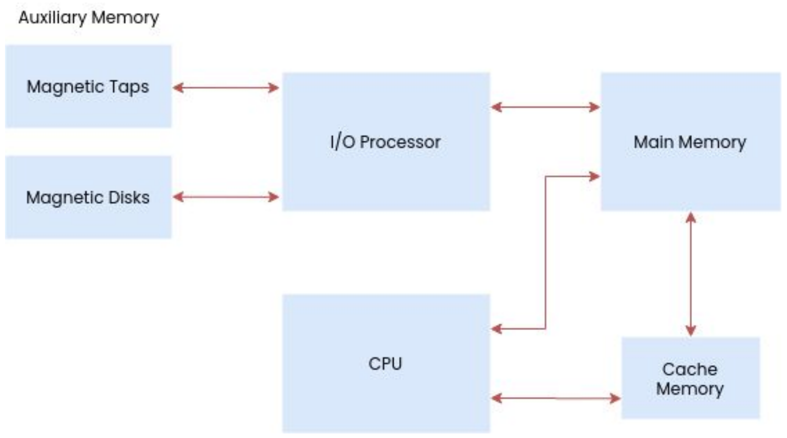
teachics.org

Computer Organization & Architecture

Module 5 - Part 1

Memory Organization

Memory Hierarchy



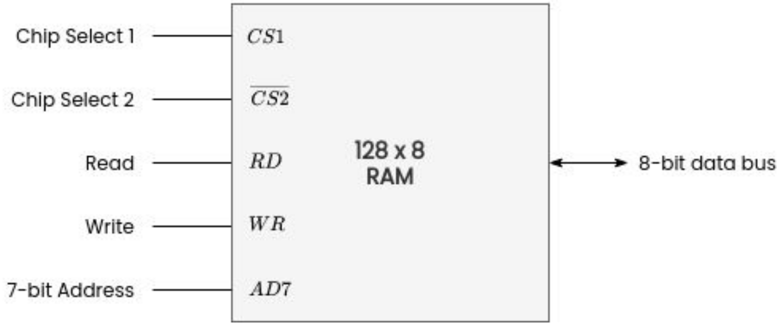
- The memory unit that communicates directly with the CPU is called the **main memory**.
- Devices that provide backup storage are called **auxiliary memory**(magnetic disks and tapes).
- **Cache** is a very-high-speed memory used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate.
- Cache compensates the speed differential between main memory access time and processor logic.

Main Memory

- Most of the main memory in a general-purpose computer is made up of RAM integrated circuit chips, but a portion of the memory may be constructed with ROM chips.
- Integrated circuit **RAM(Random Access Memory)** chips are available in two possible operating modes ,Static and Dynamic.
- **Static RAM** - consists of internal flip-flops that store the binary information.
  The stored information remains valid as long as power is applied to the unit.
  Easier to use and has shorter read and write cycles.
- **Dynamic RAM** - stores binary information in the form of electric charges that are applied to capacitors.
  The stored charge on the capacitors discharge with time and the capacitors must be periodically recharged by refreshing the dynamic memory.
  Reduced power consumption and larger storage capacity in a single memory chip.
- **ROM (Read Only Memory)** is used for storing programs that are permanently resident in the computer that do not change in value once the production of the computer is completed.
- ROM portion of main memory is needed for storing an initial program called a **bootstrap loader**.

RAM Chip

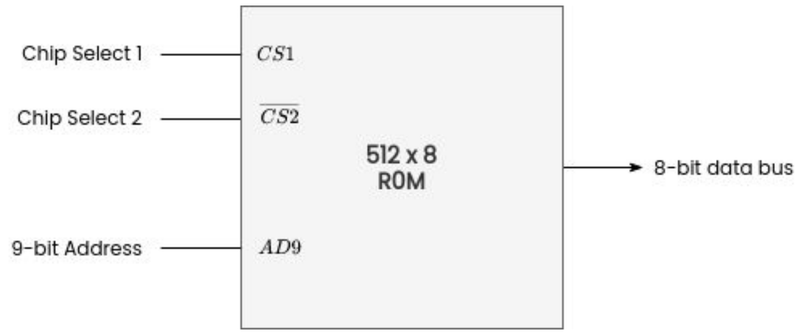
- Bidirectional data bus that allows the transfer of data either from memory to CPU(read operation) or from CPU to memory(write operation).
- The capacity of the memory is 128 words of eight bits per word. This requires a 7-bit address and an 8-bit bidirectional data bus.
- The read and write inputs specify the memory operation.
- The two chip select (CS) control inputs are for enabling the chip only when it is selected by the processor.
- The function table specifies the operation of the RAM chip.



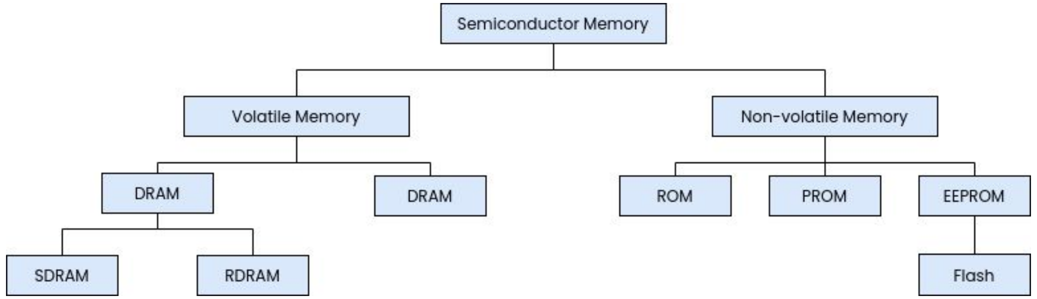
$CS1$	$\overline{CS2}$	RD	WR	*Memory Function*	*State of databus*
0	0	X	X	Inhibit	High-impedance
0	1	X	X	Inhibit	High-impedance
`1	0	0	0	Inhibit	High-impedance
1	0	0	1	Write	Input data to RAM
1	0	1	X	Read	Output data from RAM
1	1	X	X	Inhibit	High-impedance

ROM Chip

- ROM can only read, the data bus can only be in an output mode.
- For the same size chip, it is possible to have more bits of ROM than of RAM, because the internal binary cells in ROM occupy less space than in RAM.
- For this reason, the diagram specifies a 512 byte ROM, while the RAM has only 128 bytes.
- The nine address lines in the ROM chip specify any one of the 512 bytes stored in it.



Memory Technology



<u>ROM</u>

- ROMs permanently stores bit values.
- At the time of manufacture, a diode is present at the intersection of a data line and an address line.
- In a memory cell if a diode is present, the diode starts conducting and therefore the corresponding data line become high.
- If a diode is not present, the corresponding data line become low.
- Thus whether a 0 or 1 is stored in a memory cell is determined by the presence of absence of a diode.
- The data that are stored in a ROM chip is programmed at production and no later changes are possible.

Memory Technology

PROM

- Structurally similar to ROM.
- At every intersection there is a fuse.
- A charge sent through a column will pass through the fuse in a memory cell to the bit line indicating a value of 1.
- Initial state of PROM chip is all 1s.
- A PROM writing device is used to inject current flow through the required fuses.
- Unlike ROMs PROMs can be programmed once.

EEPROM

- At every intersection there is a transistor(floating-gate-metal-oxide-semiconductor(FGMOS)).
- When a high voltage is applied to a transistor, the electrons accumulate on the floating gate(avalanche injection).
- These trapped electrons continue to be present even when the high voltage is removed.
- Data stored in EEPROM can therefore be retained even for decades.

Memory Technology

Flash Memory

- Memory cells are made from floating-gate MOSFETs.
- It can either write or erase and write multi-kilobyte of data in a single operation.
- Access time is several hundred times faster than EEPROM.
- Two types- NOR Flash, NAND Flash.

SRAM

- These are memories that consist of circuits capable of retaining their state as long as power is on.
- Uses 6 transistors to form a special flip-flop circuit that can store a logic 1 or 0.
- Expensive.
- Faster than DRAM.

DRAM

- The main memory is generally made up of DRAM chips.
- Information is stored in a DRAM cell in the form of a charge on a capacitor.
- This charge needs to be periodically recharged.

Memory Technology

SDRAM

- These RAM chips' access speed is directly synchronized with the CPU's clock.
- For this, the memory chips remain ready for operation when the CPU expects them to be ready.
- Data transfer takes place both in the falling and rising edges of the clock(faster operation).
- DDR SDRAM is a faster version of SDRAM.

RDRAM (Rambus DRAM)

- It provides a very high data transfer rate over a narrow CPU-memory bus.
- It uses various speedup mechanisms, like synchronous memory interface, caching inside the DRAM chips and very fast signal timing.
- The Rambus data bus width is 8 or 9 bits.

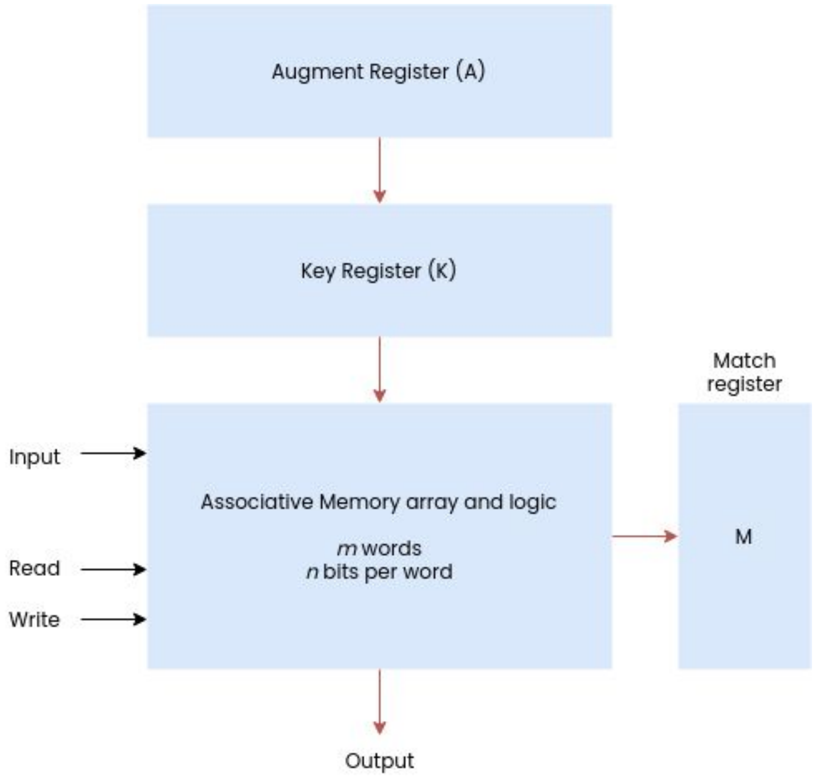
Associative Memory

Associative Memory

- The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address.
- A memory unit accessed by content is called an associative memory or **content addressable memory (CAM)**.
- This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location.
- When a word is written in an associative memory, no address is given.
- The memory is capable of finding an empty unused location to store the word.
- When a word is to be read from an associative memory, the content of the word, or part of the word, is specified.
- The memory locates all words which match the specified content and marks them for reading.
- An associative memory is more expensive than a random access memory because each cell must have storage capability as well as logic circuits for matching its content with an external argument.
- For this reason, associative memories are used in applications where the search time is very critical and must be very short.

Hardware Organization

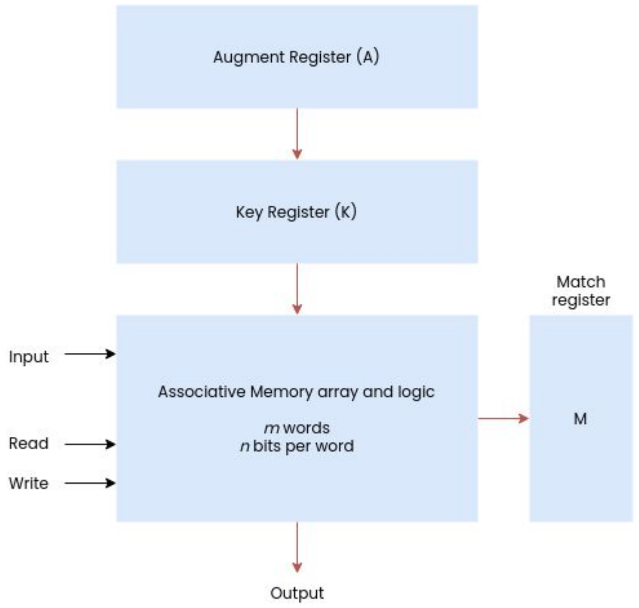
- It consists of a memory array and logic for m words with n bits per word.
- The argument register A and key register K each have n bits, one for each bit of a word.
- The match register M has m bits, one for each memory word.
- Each word in memory is compared in parallel with the content of the argument register.
- The words that match the bits of the argument register set a corresponding bit in the match register.
- After the matching process, those bits in the match register that have been set indicate the fact that their corresponding words have been matched.



Hardware Organization

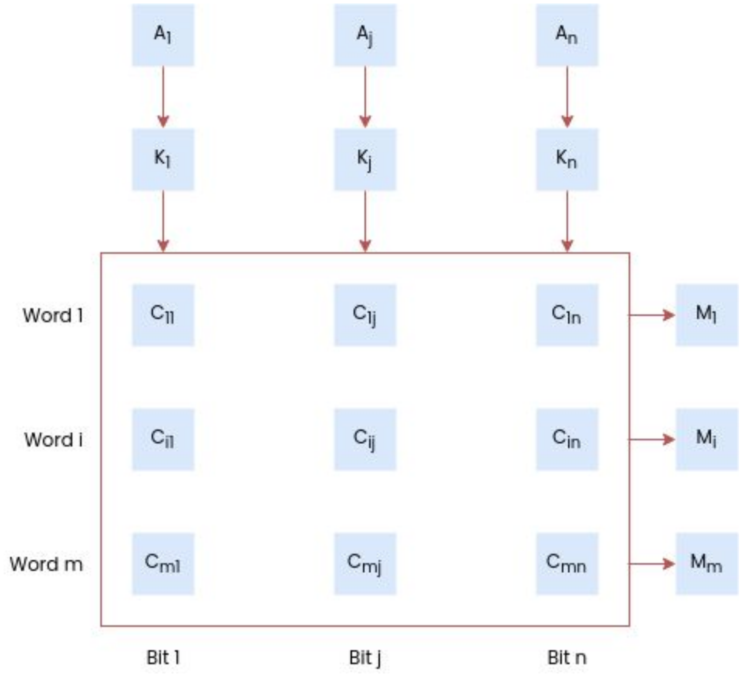
- Reading is accomplished by a sequential access to memory for those words whose corresponding bits in the match register have been set.
- The key register provides a mask for choosing a particular field or key in the argument word.
- The entire argument is compared with each memory word if the key register contains all 1's.
- Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared.
- In the following example, only the three leftmost bits of A are compared with memory words.

A	**101** 111100
K	**111** 000000
Word 1	100 111100 - no match
Word 2	101 000001 - match



Associative memory of m word, n cells per word

- The relation between the memory array and external registers in an associative memory is shown in Figure.
- The cells in the array are marked by the letter C with two subscripts.
- The first subscript gives the word number and the second specifies the bit position in the word.
- Thus cell Cij is the cell for bit j in word i.
- A bit Aj in the argument register is compared with all the bits in column j of the array provided that Kj= 1.
- This is done for all columns j = 1, 2, . . . , n.
- If a match occurs between all the unmasked bits of the argument and the bits in word i, the corresponding bit Mi in the match register is set to 1.
- If one or more unmasked bits of the argument and the word do not match, Mi is cleared to 0.



Cache Memory

Locality of Reference

- Cache memory is based on the principle of locality of reference.
- **Locality of reference**

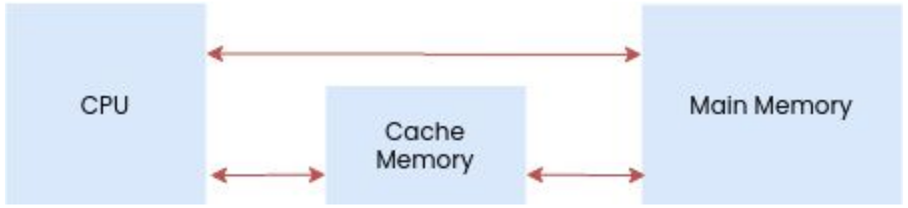
  Locality of reference refers to a phenomenon in which a computer program tends to access same set of memory locations for a particular time period.
- **Temporal locality**

  Temporal locality means current data or instruction that is being fetched may be needed soon.
- **Spatial locality**

  Spatial locality means instruction or data near to the current memory location that is being fetched, may be needed soon in the near future.
- If the active portions of the program and data are placed in a fast small memory, the total execution time of the program can be reduced.

Cache Memory

- A fast small memory placed between the CPU and main memory is referred to as a **cache memory**.



- The cache memory access time is less than the access time of main memory by a factor of 5 to 10.
- The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components.
- The fundamental idea of cache organization is that by keeping the most frequently accessed instructions and data in the fast cache memory, the average memory access time will approach the access time of the cache.
- Although the cache is only a small fraction of the size of main memory, a large fraction of memory requests will be found in the fast cache memory because of the locality of reference property of programs.

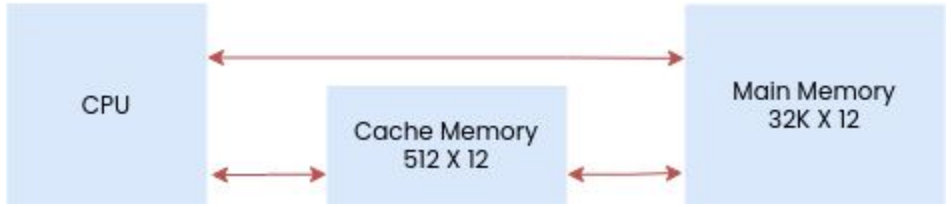
Operation & Performance

- When the CPU needs to access memory, the cache is examined.
- If the word is found in the cache, it is read from the fast memory.
- If the word addressed by the CPU is not found in the cache, the main memory is accessed to read the word.
- A block of words containing the one just accessed is then transferred from main memory to cache memory.
- The block size may vary from one word (the one just accessed) to about 16 words adjacent to the one just accessed.
- In this manner, some data are transferred to cache so that future references to memory find the required words in the fast cache memory.

- The performance of cache memory is frequently 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 and it counts as a **miss**.
- The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the **hit ratio**.
- Hit Ratio = Number of hits / (Number of hits + Number of misses)

Mapping

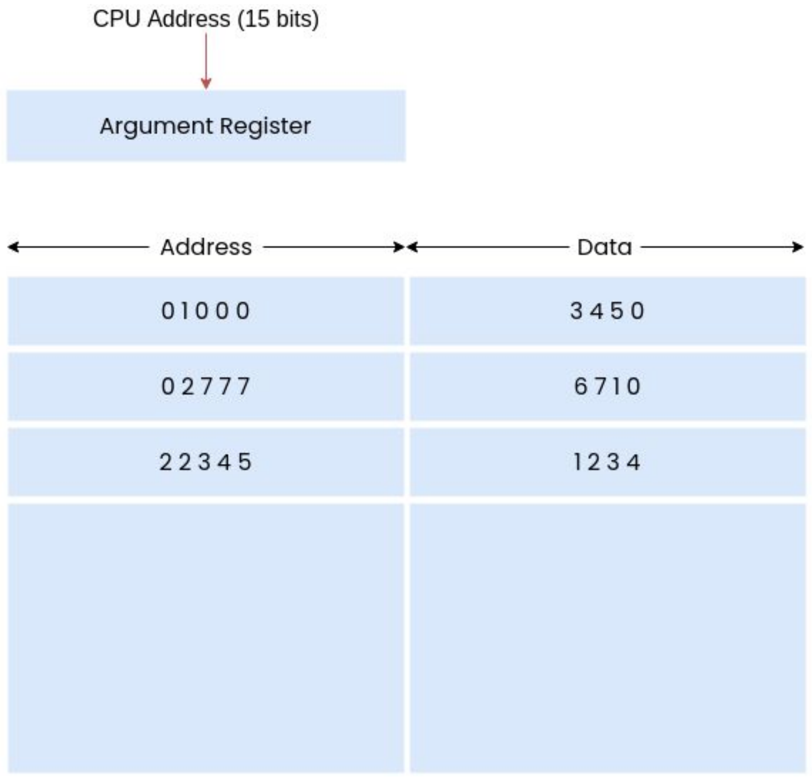
- The transformation of data from main memory to cache memory is referred to as a **mapping** process.
- Three types of mapping procedures
  1. Associative mapping
  2. Direct mapping
  3. Set-associative mapping



- A specific example of a memory organization as shown in Figure is used for discussing three mappings.
- The main memory can store 32K words of 12 bits each.
- The cache is capable of storing 512 of these words at any given time.
- The CPU communicates with both memories.

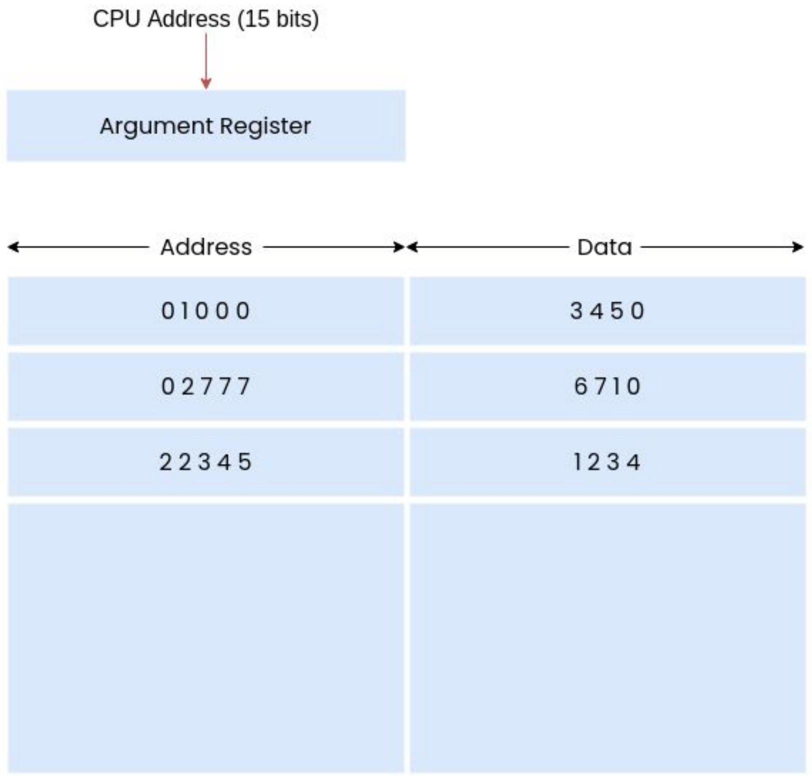
Associative Mapping

- The fastest and most flexible cache organization uses an associative memory.
- The associative memory stores both the address and content (data) of the memory word.
- This permits any location in cache to store any word from main memory.
- The diagram shows three words presently stored in the cache.
- The address value of 15 bits is shown as a five-digit octal number and its corresponding 12-bit word is shown as a four-digit octal number.
- A CPU address of 15 bits is placed in the argument register and the associative memory is searched for a matching address.



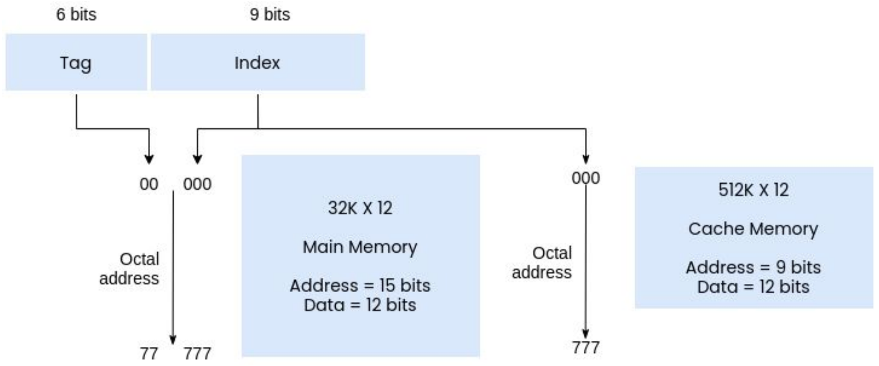
Associative Mapping

- If the address is found, the corresponding 12-bit data is read and sent to the CPU.
- If no match occurs, the main memory is accessed for the word. The address—data pair is then transferred to the associative cache memory.
- If the cache is full, an address—data pair must be displaced to make room for a pair that is needed and not presently in the cache.
- Several replacement policies like first-in first-out (FIFO) is used for this.



Direct Mapping

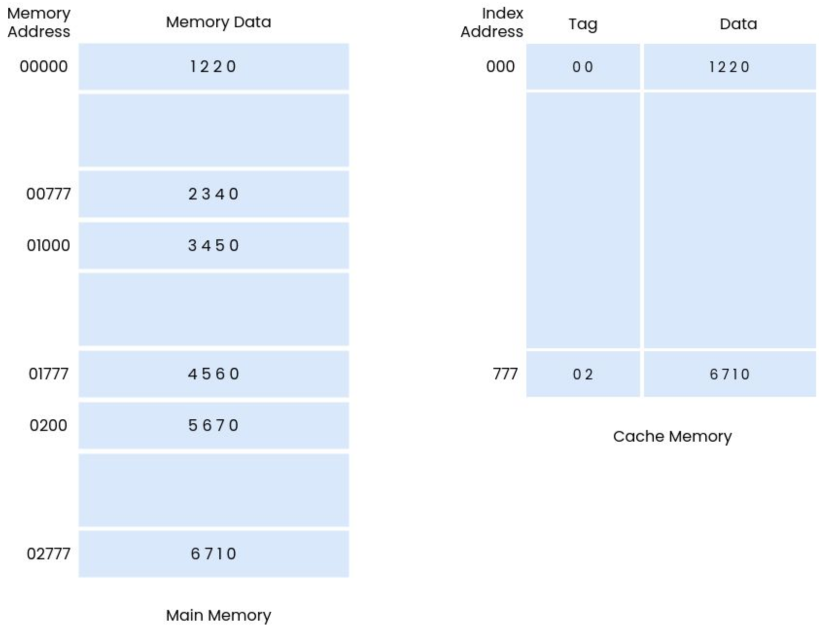
- Associative memories are expensive compared to random-access memories because of the added logic associated with each cell.
- It uses a random-access memory for the cache.
- The CPU address of 15 bits is divided into two fields.
- The nine least significant bits constitute the index field and the remaining six bits form the tag field.
- The main memory needs an address that includes both the tag and the index bits.



- The number of bits in the index field is equal to the number of address bits required to access the cache memory.
- In the general case, there are 2^k words in cache memory and 2^n words in main memory.
- The n-bit memory address is divided into two fields: k bits for the index field and n k bits for the tag field.
- The direct mapping cache organization uses the n-bit address to access the main memory and the k-bit index to access the cache.

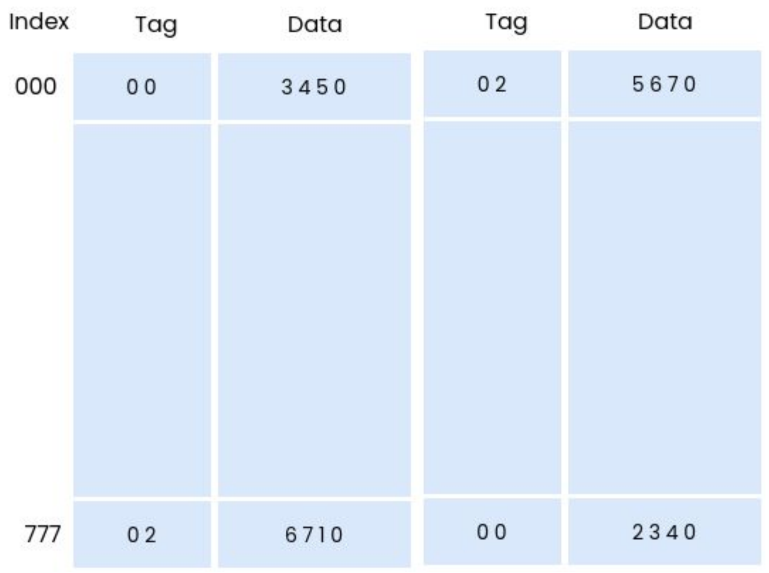
Direct Mapping

- Each word in cache consists of the data word and its associated tag.
- When a new word is first brought into the cache, the tag bits are stored alongside the data bits.
- When the CPU generates a memory request, the index field is used for the address to access the cache.
- The tag field of the CPU address is compared with the tag in the word read from the cache.
- If they match - hit - the desired data word is in cache.
- If they don't match - miss - required word is read from main memory.
- It is then stored in the cache together with the new tag, replacing the previous value.
- Disadvantage - the hit ratio can drop considerably if two or more words whose addresses have the same index but different tags are accessed repeatedly.



Set Associative Mapping

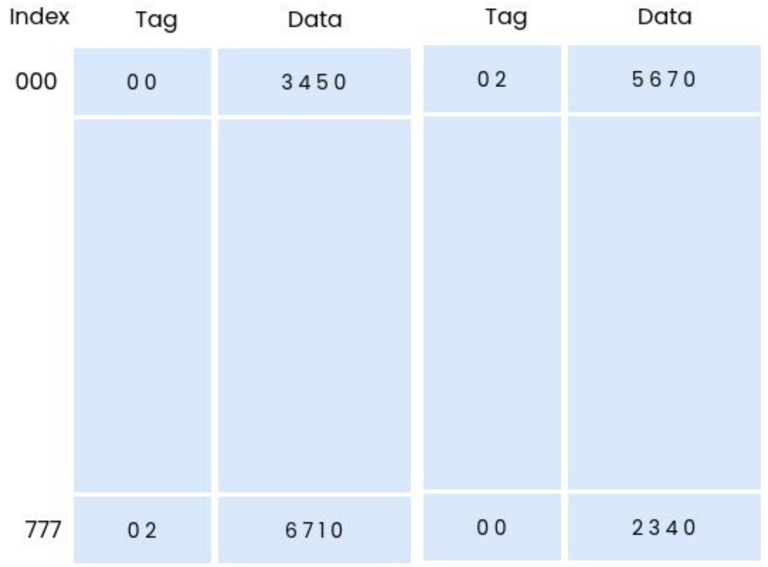
- Set-associative mapping, is an improvement over the direct mapping organization in that each word of cache can store two or more words of memory under the same index address.
- Each data word is stored together with its tag and the number of tag—data items in one word of cache is said to form a set.
- An example of a set-associative cache organization for a set size of two is shown in Figure.
- Each index address refers to two data words and their associated tags.
- A set-associative cache of set size k will accommodate k words of main memory in each word of cache.
-



Index	Tag	Data	Tag	Data
000	0 0	3 4 5 0	0 2	5 6 7 0
777	0 2	6 7 1 0	0 0	2 3 4 0

Set Associative Mapping

- When the CPU generates a memory request, the index value of the address is used to access the cache.
- The tag field of the CPU address is then compared with both tags in the cache to determine if a match occurs.
- The comparison logic is done by an associative search of the tags in the set similar to an associative memory search: thus the name “set-associative.”
- The hit ratio will improve as the set size increases Because more words with the same index but different tags can reside in cache.
- When a miss occurs in a set-associative cache and the Set is full, it is necessary to replace one of the tag-data items with a new value.
- The most common replacement algorithms used are: random replacement, first-in, first-out (FIFO), and least recently used (LRU).



Index	Tag	Data	Tag	Data
000	00	3450	02	5670
777	02	6710	00	2340

Writing into Cache

- The simplest and most commonly used procedure is to update main memory with every memory write operation, with cache memory being updated in parallel if it contains the word at the specified address.
- This is called the **write-through** method.
- The second procedure is called the **write-back method**.
- In this method only the cache location is updated during a write operation.
- The location is then marked by a flag so that later when the word is removed from the cache it is copied into main memory.

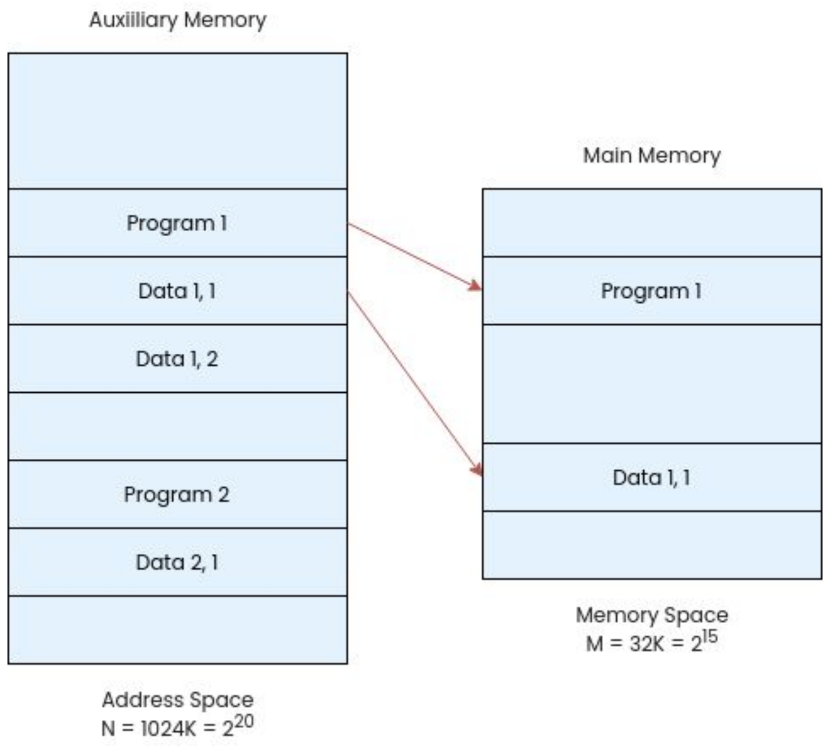
Virtual Memory

Virtual Memory

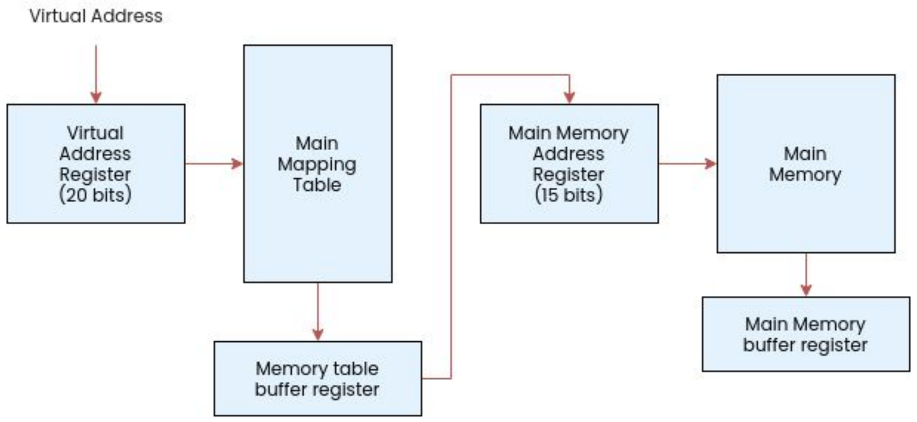
- In a memory hierarchy system, programs and data are first stored in auxiliary memory.
- Portions of a program or data are brought into main memory as they are needed by the CPU.
- Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory.
- Each address that is referenced by the CPU goes through an address mapping from the so-called virtual address to a physical address in main memory.
- Virtual memory is used to give programmers the illusion that they have a very large memory at their disposal, even though the computer actually has a relatively small main memory.
- A virtual memory system provides a mechanism for translating program-generated addresses into correct main memory locations.
- This is done dynamically, while programs are being executed in the CPU.
- The translation or mapping is handled automatically by the hardware by means of a mapping table.

Address Space and Memory Space

- An address used by a programmer will be called a **virtual address**, and the set of such addresses the **address space**.
- An address in main memory is called a location or **physical address**. The set of such locations is called **memory space**.
- Consider a computer with a main-memory capacity of 32K words (K 1024).
- 15 bits are needed to specify a physical address in memory since 32K=2^{15}.
- Suppose that the computer has available auxiliary memory for storing 2^{20}=1024K words.
- Denoting the address space by N and the memory space by M, we then have for this example N=1024K and M=32K.



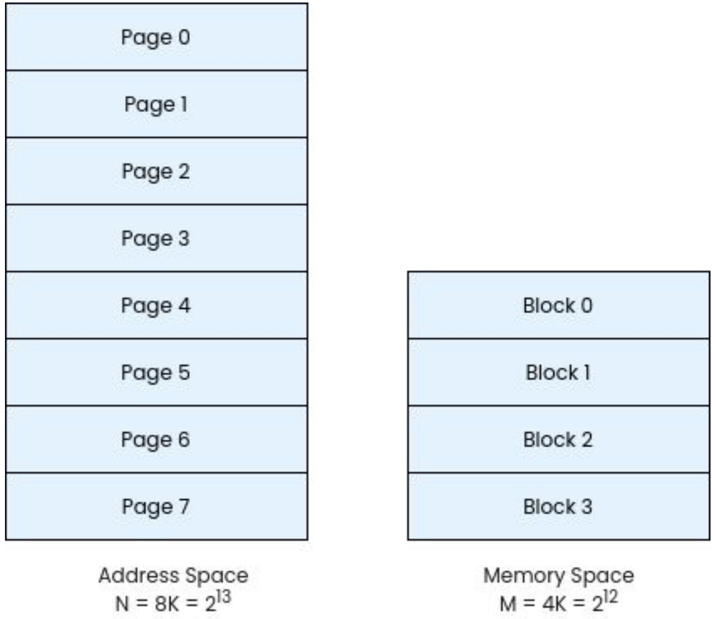
Virtual Memory Mapping


Virtual Address
Virtual Address Register (20 bits)
Main Mapping Table
Memory table buffer register
Main Memory Address Register (15 bits)
Main Memory
Main Memory buffer register

- Virtual Address (20 bits)
- Physical Address (15 bits)

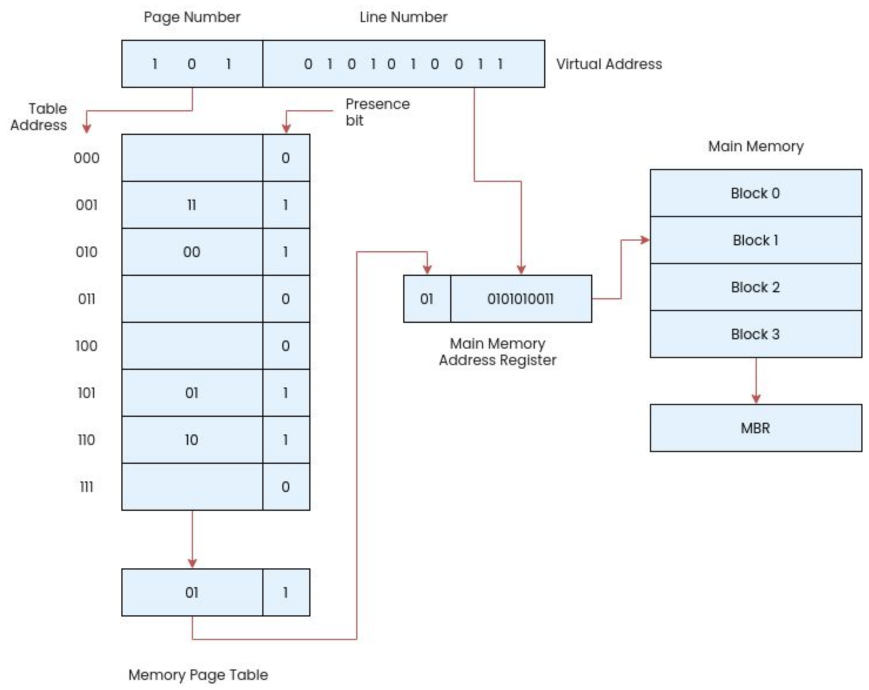
Address Mapping Using Pages

- 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.
- Each **virtual address** is represented by two numbers: a **page number** address and a **line** within the page.
- In a computer with 2^P words per page, p bits specify a line address and the remaining bits specify the page number.
- 10 bit line number, 3 bit page number.
- Line address in address space and memory space is the same.
- Mapping is only required from a page number to a block number.



Memory Mapping Table

- The memory-page table consists of eight words, one for each page.
- The address in the page table denotes the page number and the content of the word gives the block number where that page is stored in main memory.
- A presence bit in each location indicates whether the page has been transferred from auxiliary memory into main memory.
- The CPU references a word in memory with a virtual address of 13 bits.
- The three high-order bits of the virtual address specify a page number and also an address for the memory-page table.



Memory Mapping Table

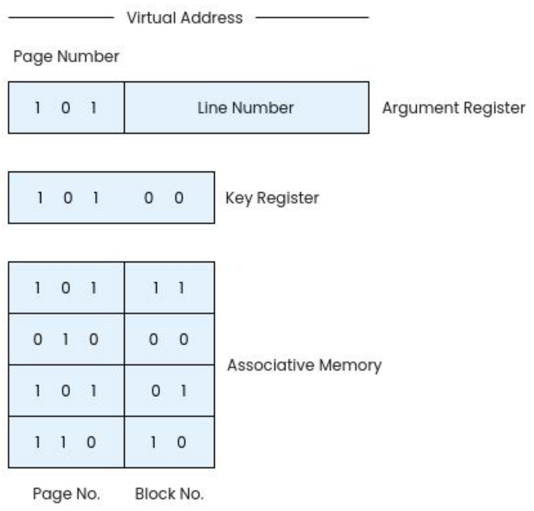
- The content of the word in the memory page table at the page number address is read out into the memory table buffer register.
- If the presence bit is a 1, the block number thus read is transferred to the two high-order bits of the main memory address register.
- The line number from the virtual address is transferred into the 10 low-order bits of the memory address register.
- A read signal to main memory transfers the content of the word to the main memory buffer register ready to be used by the CPU.
- If the presence bit in the word read from the page table is 0, it signifies that the content of the word referenced by the virtual address does not reside in main memory.
- A call to the operating system is then generated to fetch the required page from auxiliary memory and place it into main memory before resuming computation.

Associative Memory Page Table

- A random-access memory page table is inefficient with respect to storage utilization.
- A system with n pages and m blocks would require a memory-page table of n locations of which up to m blocks will be marked with block numbers and all others will be empty.
- Consider an address space of 1024K words and memory space of 32K words.
- If each page or block contains 1K words, the number of pages is 1024 and the number of blocks 32.
- The capacity of the memory-page table must be 1024 words and only 32 locations may have a presence bit equal to 1.
- At any given time, at least 992 locations will be empty and not in use.
- A more efficient way to organize the page table would be to construct it with a number of words equal to the number of blocks in main memory.
- In this way the size of the memory is reduced and each location is fully utilized.
- This method can be implemented by means of an associative memory with each word in memory containing a page number together with its corresponding block number.
- The page field in each word is compared with the page number in the virtual address.
- If a match occurs, the word is read from memory and its corresponding block number is extracted.

Associative Memory Page Table

- We replace the random access memory-page table with an associative memory of four words.
- Each entry in the associative memory array consists of two fields.
- The first three bits specify a field for storing the page number.
- The last two bits constitute a field for storing the block number.
- The virtual address is placed in the argument register.
- The page number bits in the argument register are compared with all page numbers in the page field of the associative memory.
- If the page number is found, the 5-bit word is read out from memory.
- The corresponding block number, being in the same word, is transferred to the main memory address register.
- If no match occurs, a call to the operating system is generated to bring the required page from auxiliary memory



Page Replacement

- When a program starts execution, one or more pages are transferred into main memory and the page table is set to indicate their position.
- The program is executed from main memory until it attempts to reference a page that is still in auxiliary memory.
- This condition is called **page fault**.
- When page fault occurs, the execution of the present program is suspended until the required page is brought into main memory.
- When a page fault occurs a new page is then transferred from auxiliary memory to main memory.
- If main memory is full, a page is selected and removed from a memory block using some replacement algorithm.
- Two of the most common replacement algorithms used are the first-in, first-out (FIFO) and the least recently used (LRU).
- The LRU policy removes the least recently used page is removed.
- LRU - The least recently used page is the page with the highest count. The counters are often called aging registers, as their count indicates their age, that is, how long ago their associated pages have been referenced.

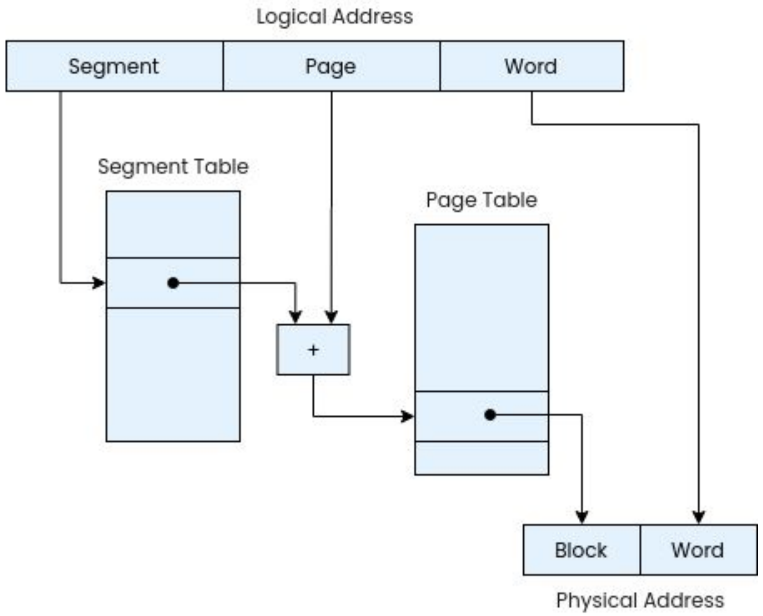
Memory Management Hardware

Memory Management Hardware

- A memory management system is a collection of hardware and software procedures for managing the various programs residing in memory.
- The basic components of a memory management unit are:
  1. A facility for dynamic storage relocation that maps logical memory references into physical memory addresses.
  2. A provision for sharing common programs stored in memory by different users.
  3. Protection of information against unauthorized access between users and preventing users from changing operating system functions.
- The dynamic storage relocation hardware is a mapping process similar to the paging system in virtual memory.
- For convenience programs and data are divided into logical parts called segments.
- A **segment** is a set of logically related instructions or data elements associated with a given name.
- The address generated by a segmented program is called a **logical address**.
- This is similar to a virtual address except that logical address space is associated with variable-length segments rather than fixed-length pages.

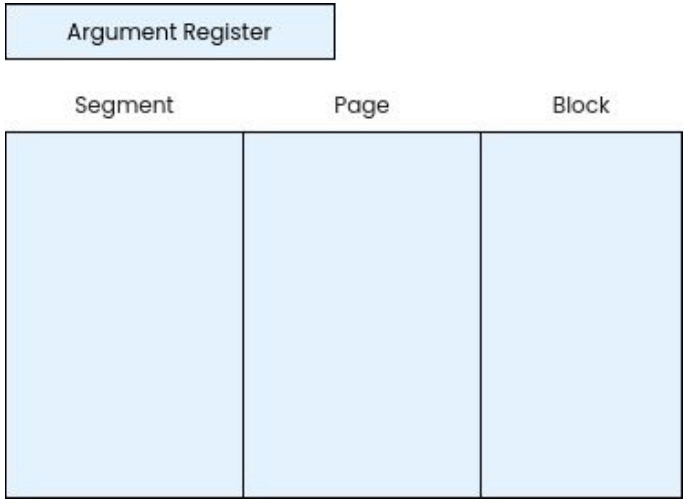
Segmented-Page Mapping

- The logical address is partitioned into three fields.
- The segment field specifies a segment number.
- The page field specifies the page within the segment,
- The word field gives the specific word within the page.
- The mapping of the logical address into a physical address is done by means of two tables.
- The segment number of the logical address specifies the address for the segment table.
- The entry in the segment table is a pointer address for a page table base.
- The page table base is added to the page no produces a pointer address to an entry in the page table.
- The value found in the page table provides the block number in physical memory



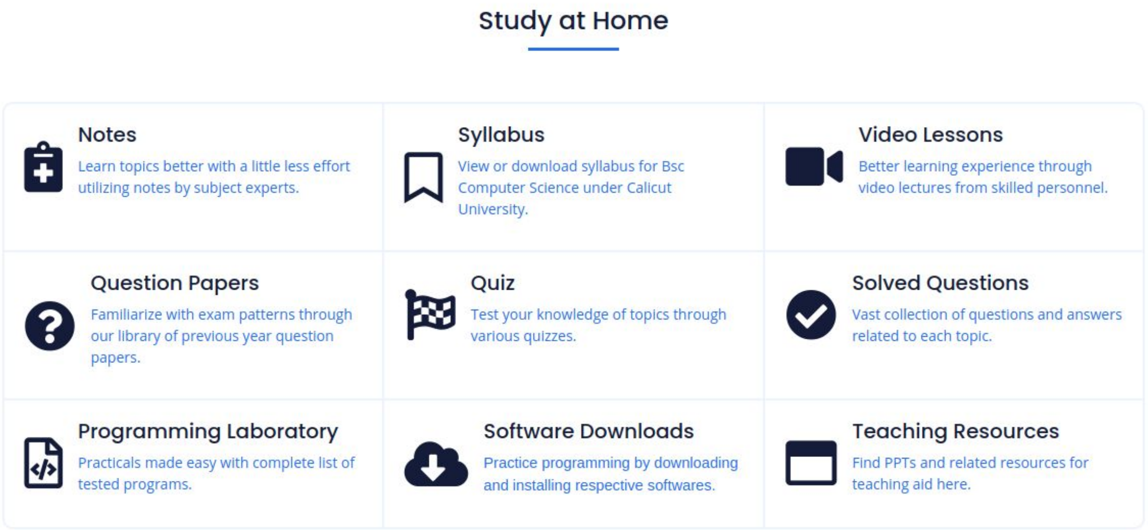
Translation Lookaside Buffer

- The two mapping tables may be stored in two separate small memories or in main memory.
- In either case, a memory reference from the CPU will require three accesses to memory: (from the segment table, from the page table, and from main memory.)
- To avoid the speed penalty, a fast associative memory is used to hold the most recently referenced table entries.
- This type of memory is sometimes called a translation lookaside buffer, abbreviated TLB.



Memory Protection

- Memory protection can be assigned to the physical address or the logical address.
- The protection of memory through the physical address can be done by assigning to each block in memory a number of protection bits that indicate the type of access allowed to its corresponding block.
- A much better place to apply protection is in the logical address space rather than the physical address space.
- This can be done by including protection information within the segment table or segment register of the memory management hardware.
- The protection field in a segment descriptor specifies the access rights available to the particular segment.
- In a segmented-page organization, each entry in the page table may have its own protection field to describe the access rights of each page.
- Some of the access rights of interest that are used for protecting the programs residing in memory are:
  1. Full read and write privileges
  2. Read only (write protection)
  3. Execute only (program protection)
  4. System only (operating system protection



Study at Home

Notes Learn topics better with a little less effort utilizing notes by subject experts.	**Syllabus** View or download syllabus for Bsc Computer Science under Calicut University.	**Video Lessons** Better learning experience through video lectures from skilled personnel.
Question Papers Familiarize with exam patterns through our library of previous year question papers.	**Quiz** Test your knowledge of topics through various quizzes.	**Solved Questions** Vast collection of questions and answers related to each topic.
Programming Laboratory Practicals made easy with complete list of tested programs.	**Software Downloads** Practice programming by downloading and installing respective softwares.	**Teaching Resources** Find PPTs and related resources for teaching aid here.

www.teachics.org

teachics.org

Computer Organization & Architecture

Module 5 - Part 2

Contents

Input-Output Organization

I/O Interface

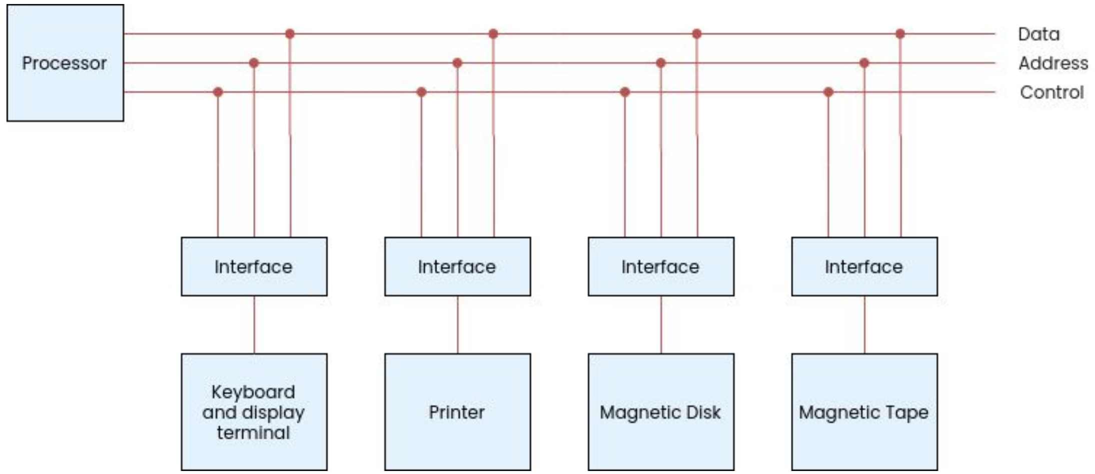
Need for I/O Interface

- Input–output interface provides a method for transferring information between internal storage and external I/O devices.
- Peripherals connected to a computer need special communication links for interfacing them with the CPU and each peripheral.
- The purpose of the communication link is to resolve the differences that exist between the central computer and each peripheral.
- The major differences are:
  1. Peripherals are electromechanical and electromagnetic devices and their manner of operation is different from the operation of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be required.
  2. The data transfer rate of peripherals is usually slower than the transfer rate of the CPU, and consequently, a synchronization mechanism may be needed.
  3. Data codes and formats in peripherals differ from the word format in the CPU and memory.
  4. The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

I/O Interface

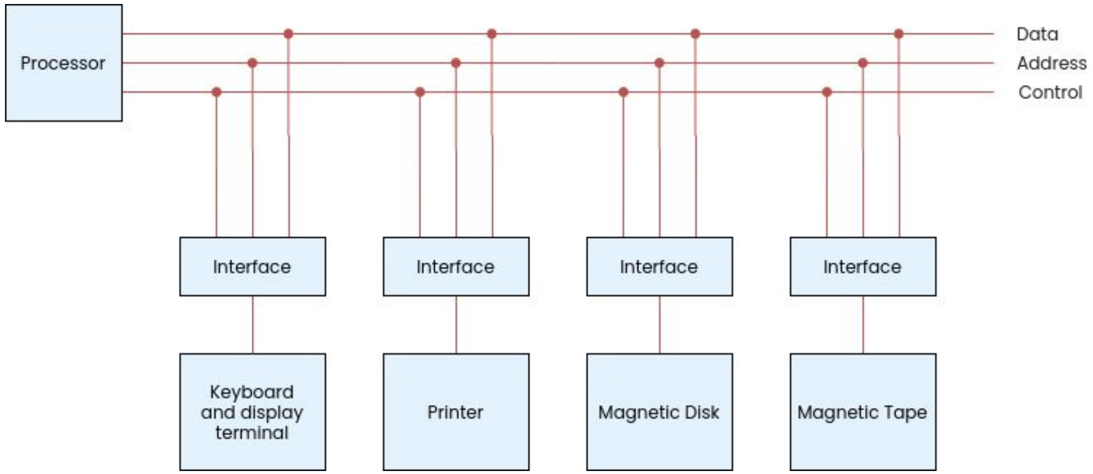
- To resolve these differences, computer systems include special hardware components between the CPU and peripherals to supervise and synchronize all input and output transfers.
- These components are called **interface** units because they interface between the processor bus and the peripheral device.
- To attach an input–output device to CPU and input–output interface, circuit is placed between the device and the system bus.
- The main function of input–output interface circuit are data conversion, synchronization and device selection.

I/O Bus and Interface Modules



- The I/O bus consists of **data** lines, **address** lines, and **control** lines.
- Each peripheral device has associated with it an **interface** unit.
- Each interface decodes the address and control received from the I/O bus, interprets them for the peripheral, and provides signals for the peripheral controller.
- It also synchronizes the data flow and supervises the transfer between peripheral and processor.

I/O Bus and Interface Modules



- Each peripheral has its own controller that operates the particular electromechanical device.
- To communicate with a particular device, the processor places a device address on the address lines.
- Each interface attached to the I/O bus contains an address decoder that monitors the address lines.
- When the interface detects its own address, it activates the path between the bus lines and the device that it controls.
- All other peripherals are disabled by their interface.

I/O Command

- At the same time that the address is made available in the address lines, the processor provides a **function code** in the control lines.
- The function code (**I/O command**) is an instruction that is executed in the interface and its attached peripheral unit.
- There are four types of commands that an interface may receive.
    1. A control command is issued to activate the peripheral and to inform it what to do.
    2. A status command is used to test various status conditions in the interface and the peripheral.
    3. A data output command causes the interface to respond by transferring data from the bus into one of its registers.
    4. The data input command allows the interface receives an item of data from the peripheral and places it in its buffer register.
- The processor checks if data are available by means of a status command and then issues a data input command.
- The interface places the data on the data lines, where they are accepted by the processor.

I/O versus Memory Bus

- In addition to communicating with I/O, the processor must communicate with the memory unit.
- Like the I/O bus, the memory bus contains data, address, and read/write control lines.
- There are three ways that computer buses can be used to communicate with memory and I/O:
  1. Use two separate buses, one for memory and the other for I/O. (IOP)
  2. Use one common bus for both memory and I/O but have separate control lines for each.(Isolated I/O)
  3. Use one common bus for memory and I/O with common control lines.(Memory Mapped I/O)

Isolated I/O

- Isolated I/O uses one common bus to transfer information between memory or I/O and the CPU.
- The distinction between a memory transfer and I/O transfer is made through separate read and write lines.
- The I/O read and I/O write control lines are enabled during an I/O transfer. The memory read and memory write control lines are enabled during a memory transfer.
- This configuration isolates all I/O interface addresses from the addresses assigned to memory and is referred to as the isolated I/O method for assigning addresses in a common bus.
- The isolated I/O method isolates memory and I/O addresses so that memory address values are not affected by interface address assignment since each has its own address space

Memory Mapped I/O

- Memory mapped I/O use the same address space for both memory and I/O.
- It employ only one set of read and write signals and do not distinguish between memory and I/O addresses.
- The computer treats an interface register as being part of the memory system.
- The assigned addresses for interface registers cannot be used for memory words, which reduces the memory address range available.
- There are no specific input or output instructions.
- The CPU can manipulate I/O data residing in interface registers with the same instructions that are used to manipulate memory words.
- The advantage is that the load and store instructions used for reading and writing from memory can be used to input and output data from I/O registers.

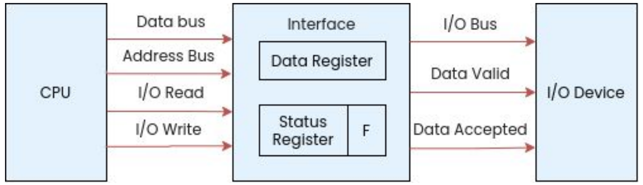
Modes Of Transfer

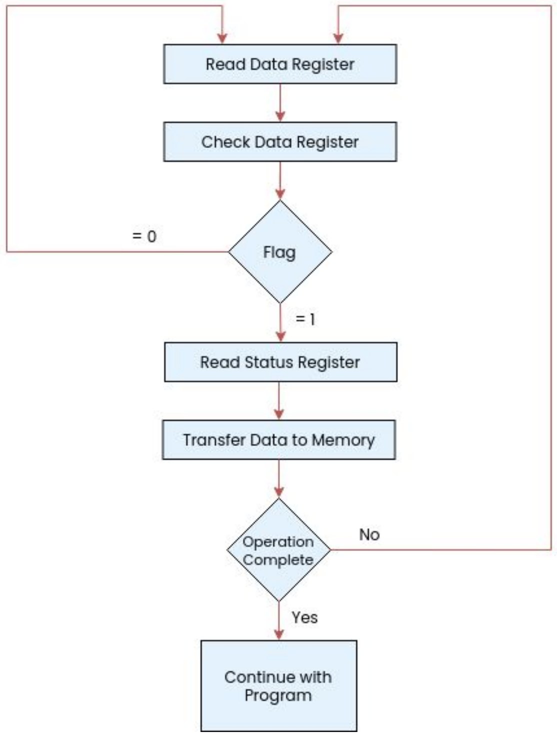
Modes Of Transfer

- Data transfer between the central computer and I/O devices may be handled in a variety of modes.
- Data transfer to and from peripherals may be handled in one of three possible modes:
  1. Programmed I/O
  2. Interrupt-initiated I/O
  3. Direct memory access (DMA)

Programmed I/O

- Programmed I/O operations are the result of **I/O instructions** written in the computer program.
- Each data item transfer is initiated by an instruction in the program.
- Transferring data under program control requires constant monitoring of the peripheral by the CPU.
- Once a data transfer is initiated, the CPU is required to monitor the interface to see when a transfer can again be made.





- **Disadvantage** : the CPU stays in a program loop until the I/O unit indicates that it is ready for data transfer. This is a time-consuming process since it keeps the processor busy needlessly.

Interrupt-Initiated I/O

- An alternative to the CPU constantly monitoring the flag is to let the interface inform the computer when it is ready to transfer data.
- This mode of transfer uses the interrupt facility.
- While the CPU is running a program, it does not check the flag.
- However, when the flag is set, the computer is momentarily interrupted from proceeding with the current program and is informed of the fact that the flag has been set.
- The CPU deviates from what it is doing to take care of the input or output transfer.
- After the transfer is completed, the computer returns to the previous program to continue what it was doing before the interrupt.
- The CPU responds to the interrupt signal by storing the return address from the program counter into a memory stack and then control branches to a service routine that processes the required I/O transfer.
- There are two methods for getting the branch address.
- One is called vectored interrupt and the other, non vectored interrupt.
- In a **nonvectored interrupt**, the branch address is assigned to a fixed location in memory.
- In a **vectored interrupt**, the interrupt source supplies the branch information to the computer. This information is called the **interrupt vector**.

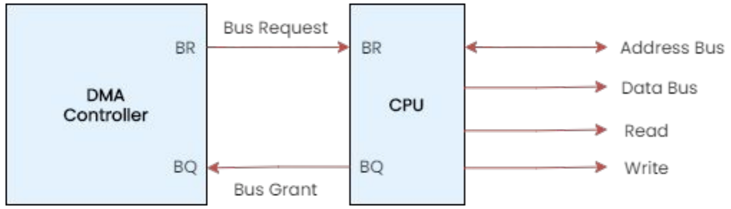
Direct Memory Access

Direct Memory Access

- In DMA, the CPU is idle and the peripheral device manage the memory buses directly.
- **DMA controller** takes over the buses to manage the transfer directly between the I/O device and memory.
- The DMA transfer can be made in several ways.
- **Burst Transfer -** A block of memory words is transferred in a continuous burst at a time. Used for fast devices such as magnetic disks, where data transmission cannot be stopped or slowed down until an entire block is transferred.
- **Cycle Stealing** - Allows the DMA controller to transfer one data word at a time, after which it must return control of the buses to the CPU. The CPU merely delays its operation for one memory cycle to allow the direct memory I/O transfer to “steal” one memory cycle.

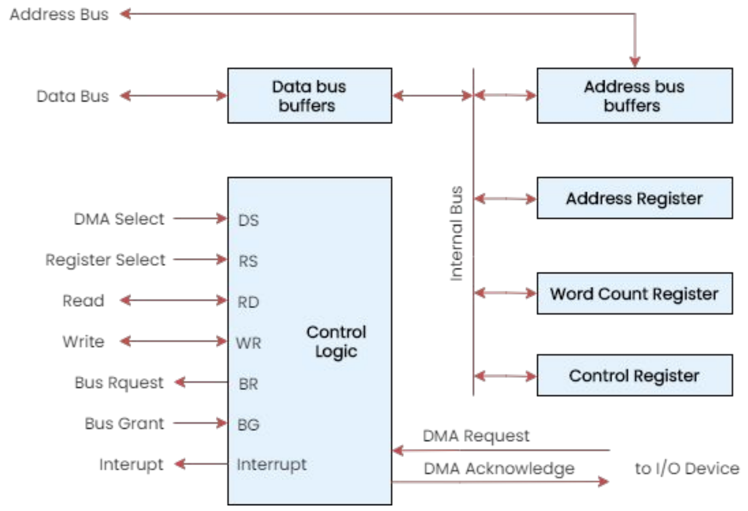
Working of DMA

- **The Bus Request (BR)** input is used by the DMA controller to request the CPU to relinquish control of the buses.
- When this input is active, the CPU terminates the execution of the current instruction and places the address bus, the data bus, and the read and write lines into a high-impedance state.(like an open circuit)
- The CPU activates the **Bus Grant (BG)** output to inform the external DMA that the buses are in the high-impedance state.(available)
- The DMA takes control of the buses to conduct memory transfers without processor intervention.
- When the DMA terminates the transfer, it disables the bus request line and the CPU disables the bus grant, takes control of the buses, and returns to its normal operation.



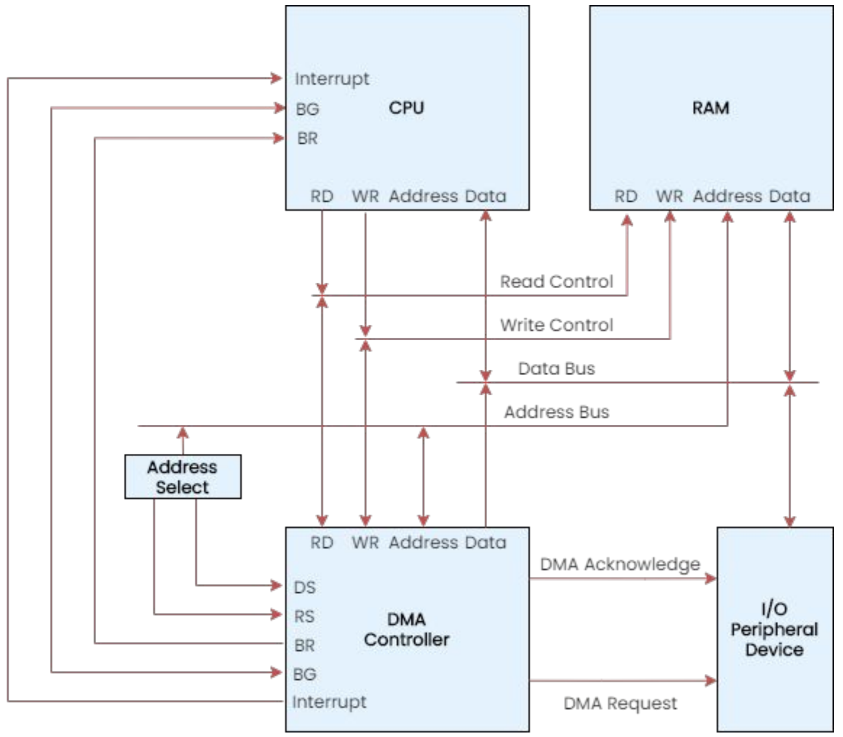
DMA Controller

- The registers in the DMA are selected by the CPU through the address bus by enabling the DMA Select (DS) and Register Select (RS) inputs.
- When the BG (bus grant) = 0, the CPU can communicate with the DMA registers.
- When BG = 1 DMA can communicate directly with the memory.
- The DMA controller has three registers.
- The **Address register** contains an address to specify the desired location in memory.
- The **Word Count Register** holds the number of words to be transferred.
- **The Control Register** specifies the mode of transfer.
- The DMA communicates with the external peripheral through the request and acknowledge lines.



DMA Transfer

- ➜ I/O Device sends a DMA request.
- ➜ DMA Controller activates the BR line.
- ➜ CPU responds with BG line.
- ➜ The DMA puts the current value of its address register into the address bus, initiates the RD or WR signal, and sends a DMA acknowledge to the I/O device.
- ➜ I/O device puts a word in the data bus (for write) or receives a word from the data bus (for read).
- ➜ The peripheral unit can then communicate with memory through the data bus for direct transfer between the two units while the CPU is momentarily disabled.



DMA Transfer

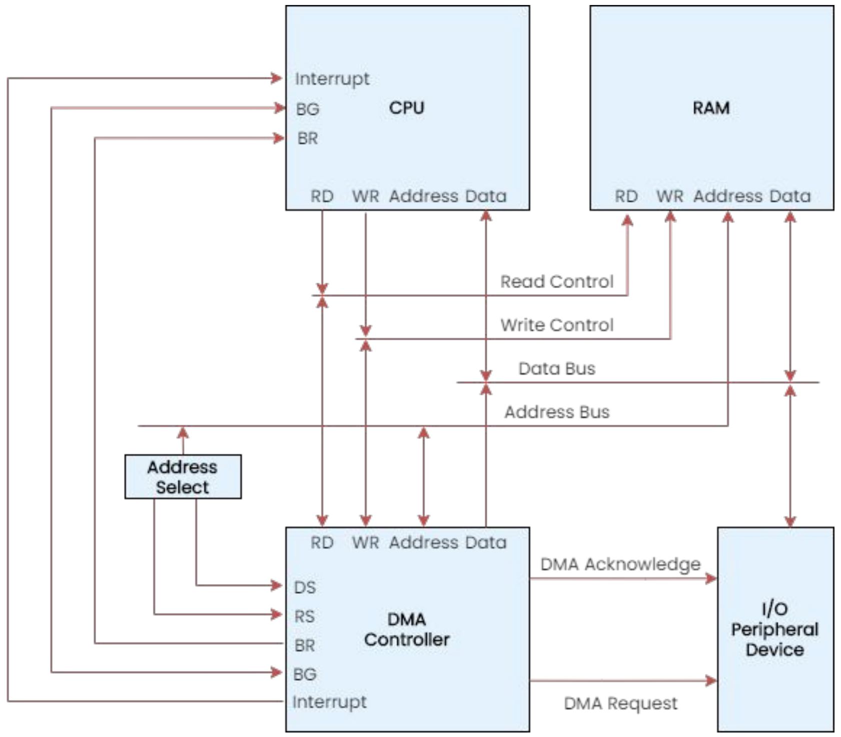
➜ For each word that is transferred, the DMA increments its address register and decrements its word count register.

➜ If the word count does not reach zero, the DMA checks the request line coming from the I/O Device.

➜ If there is no request ,the DMA disables BR so that the CPU continues to execute its own program.

➜ When CPU requests another transfer, DMA requests bus again.

➜ If the word count register reaches zero, the DMA stops any further transfer and removes its bus request. It also informs the CPU of the termination by an interrupt.



Data Transfer

Synchronous v/s Asynchronous Data Transfer

Synchronous Data Transfer

- Internal operations are synchronized by means of clock pulses.
- All data transfers among internal registers occur simultaneously during the occurrence of a clock pulse.
- The registers in the interface share a common clock with the CPU registers.

Asynchronous Data Transfer

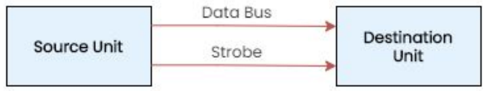
- It requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted.
- Internal timing of each unit (interface and CPU) is independent.
- Each unit uses its own private clock for internal registers.

Strobe Control and Handshaking

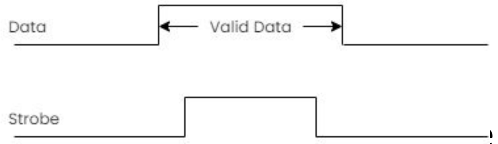
- Asynchronous data transfer between two independent units requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted.
- One way of achieving this is by means of a **strobe pulse** supplied by one of the units to indicate to the other unit when the transfer has to occur.
- 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 control signal to acknowledge receipt of the data.
- This type of agreement between two independent units is referred to as **handshaking**.
- The strobe pulse method and the handshaking method of asynchronous data transfer are not restricted to I/O transfers.

Strobe Control

- It employs a single control line to time each transfer.
- The strobe may be activated by either the source or the destination unit.
- The data bus carries the binary information from source to the destination.
- The **strobe** is a single line that informs the destinat
  data word is available in the bus.
- The source unit first places the data on the data bus.
- After a brief delay to ensure that the data settle
  source activates the strobe pulse.
- The information on the data bus and the strobe signal remain in the active state for a sufficient time period to allow the destination unit to receive the data.
- The source removes the data from the bus a brief period after it disables its strobe pulse.
- The destination unit 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 destination unit then disables the strobe.
- The source removes the data from the bus after a predetermined time interval.



(Block Diagram)



(Timing Diagram)

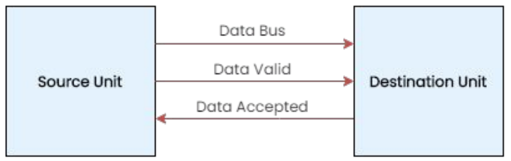
Hand Shaking

The disadvantage of the strobe method

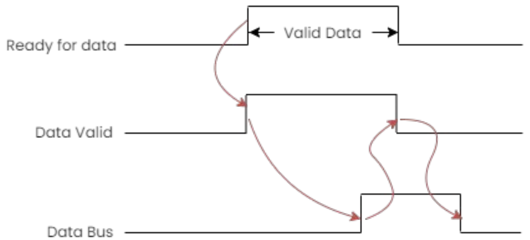
- The source unit that initiates the transfer has no way of knowing whether the destination unit has actually received the data item that was placed in the bus.
- A destination unit that initiates the transfer has no way of knowing whether the source unit has actually placed the data on the bus.

Two-wire control

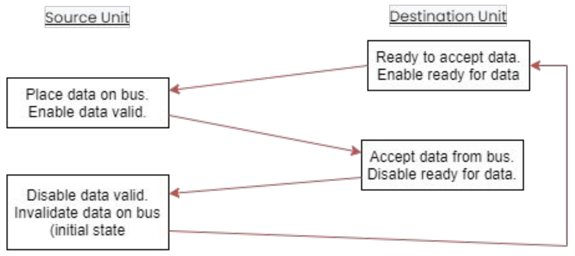
- The handshake method introduced a second control signal that provides a reply to the unit that initiates the transfer.
- One control line is in the same direction as the data flow in the bus from the source to the destination.
- It is used by the source unit to inform the destination unit whether there are valid data in the bus.
- The other control line is in the other direction from the destination to the source.
- It is used by the destination unit to inform the source whether it can accept data.
- The sequence of control during the transfer depends on the unit that initiates the transfer.
- This provides a high degree of flexibility and reliability because the successful completion of a data transfer relies on active participation by both units.
- If one unit is faulty, the data transfer will not be completed.(Detected by a time-out mechanism)



(Block Diagram)



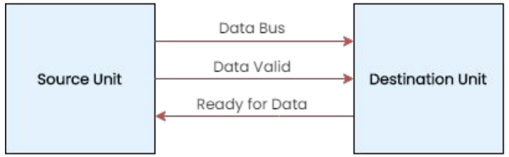
(Timing Diagram)


Source Unit
Destination Unit
Ready to accept data. Enable ready for data
Place data on bus. Enable data valid.
Accept data from bus. Disable ready for data.
Disable data valid. Invalidate data on bus (initial state

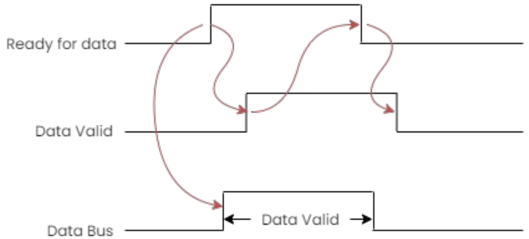
(Sequence of Events)

Source-initiated transfer using strobe control.

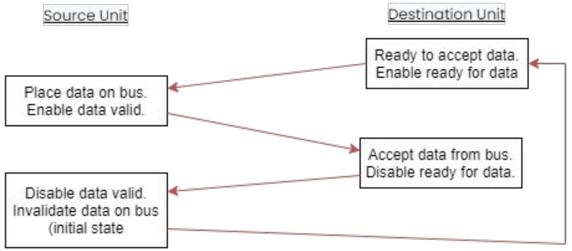
- The two handshaking lines are **data valid**(source unit) and **data accepted** (destination unit).
- The source unit initiates the transfer by placing the data on the bus and enabling its data valid signal.
- The data accepted signal is activated by the destination unit after it accepts the data from the bus.
- The source unit then disables its data valid signal, which invalidates the data on the bus.
- The destination unit then disables its data accepted signal and the system goes into its initial state.
- The source does not send the next data item until after the destination unit shows its readiness to accept new data by disabling its data accepted signal.
- This scheme allows delays from one state to the next and permits each unit to respond at its own data transfer rate.



(Block Diagram)



(Timing Diagram)


Source Unit
Destination Unit
Ready to accept data. Enable ready for data
Place data on bus. Enable data valid.
Accept data from bus. Disable ready for data.
Disable data valid. Invalidate data on bus (initial state

(Sequence of Events)

Destination-initiated transfer using strobe control.

- The source unit in this case does not place data on the bus until after it receives the ready for data signal from the destination unit.
- From there on, the handshaking procedure follows the same pattern as in the source-initiated case.
- The sequence of events in both cases would be identical if we consider the ready for data signal as the complement of data accepted.
- The only difference between the source-initiated and the destination-initiated transfer is in their choice of initial state.

Priority Interrupt

Priority Interrupt

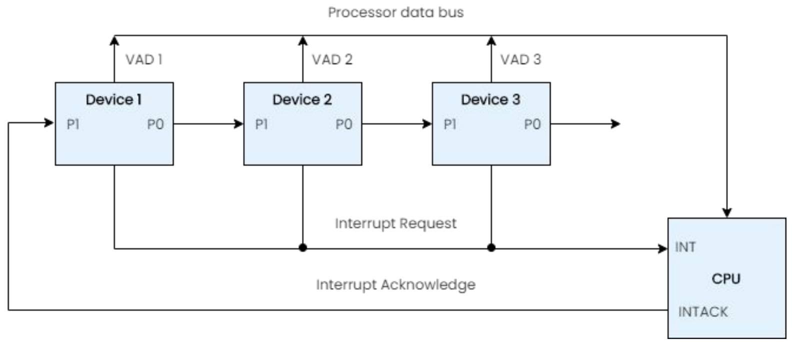
- In a typical application a number of I/O devices are attached to the computer, with each device being able to originate an interrupt request.
- The first task of the interrupt system is to identify the source of the interrupt.
- Several sources may request service simultaneously. In this case the system must also decide which device to service first.
- A **priority interrupt** is a system that establishes a priority over the various sources to determine which condition is to be serviced first when two or more requests arrive simultaneously.
- The system may also determine which conditions are permitted to interrupt the computer while another interrupt is being serviced.
- Devices with high speed transfers such as magnetic disks are given high priority, and slow devices such as keyboards receive low priority.
- When two devices interrupt the computer at the same time, the computer services the device, with the higher priority first.
- Methods:
  1. Software - Polling
  2. Hardware - Daisy chaining, Parallel Priority.

Polling

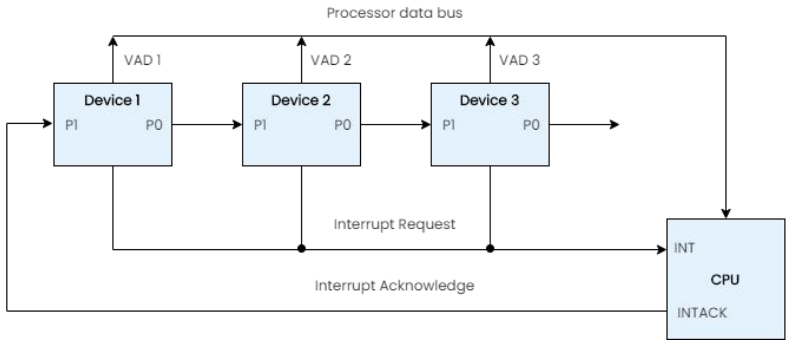
- A **polling** procedure is used to identify the highest-priority source by software means.
- In this method there is one common branch address for all interrupts.
- The program that takes care of interrupts begins at the branch address and polls the interrupt sources in sequence.
- The order in which they are tested determines the priority of each interrupt.
- The highest priority source is tested first, and if its interrupt signal is on, control branches to a service routine for this source.
- Otherwise, the next-lower-priority source is tested, and so on.
- **Disadvantage**: If there are many interrupts, the time required to poll them can exceed the time available to service the I/O device. In this situation a hardware priority-interrupt unit can be used to speed up the operation.
- **A hardware priority-interrupt** unit accepts interrupt requests from many sources, determines which of the incoming requests has the highest priority, and issues an interrupt request to the computer based on this determination.
- To speed up the operation, each interrupt source has its own interrupt vector to access its own service routine directly. Thus no polling is required.

Daisy Chaining

- The hardware priority function can be established by either a serial or a parallel connection of interrupt lines.
- The serial connection is also known as the **daisy chaining** method.
- It consists of a serial connection of all devices that request an interrupt.
- The device with the highest priority is placed in the first position, followed by lower-priority devices up to the device with the lowest priority, which is placed last in the chain.

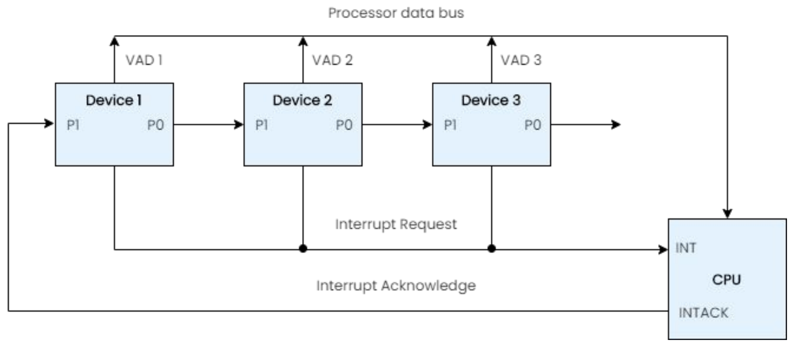


Daisy Chaining



- The interrupt request line is common to all devices and forms a wired logic connection.
- If any device has its interrupt signal in the low-level state, the interrupt line goes to the low-level state and enables the interrupt input in the CPU.
- When no interrupts arc pending, the interrupt line stays in the high-level state and no interrupts are recognized by the CPU.
- The CPU responds to an interrupt request by enabling the interrupt acknowledge line.
- This signal is received by device 1 at its PI (priority in) input.
- The acknowledge signal passes on to the next device through the PO (priority out) output only if device 1 is not requesting an interrupt.

Daisy Chaining

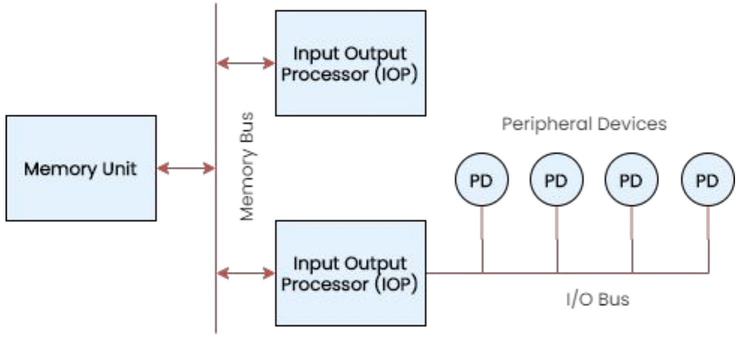


- If device 1 has a pending interrupt, it blocks the acknowledge signal from the next device by placing a 0 in the PO output.
- It then proceeds to insert its own interrupt vector address (VAD) into the data bus for the CPU to use during the interrupt cycle.
- A device with a 0 in its PI input generates a 0 in its PO output to inform the next-lower-priority device that the acknowledge signal has been blocked.
- A device that is requesting an interrupt and has a 1 in its PI input will intercept the acknowledge signal by placing a 0 in its PO output.
- If the device does not have pending interrupts, it transmits the acknowledge signal to the next device.

Input Output Processor (IOP)

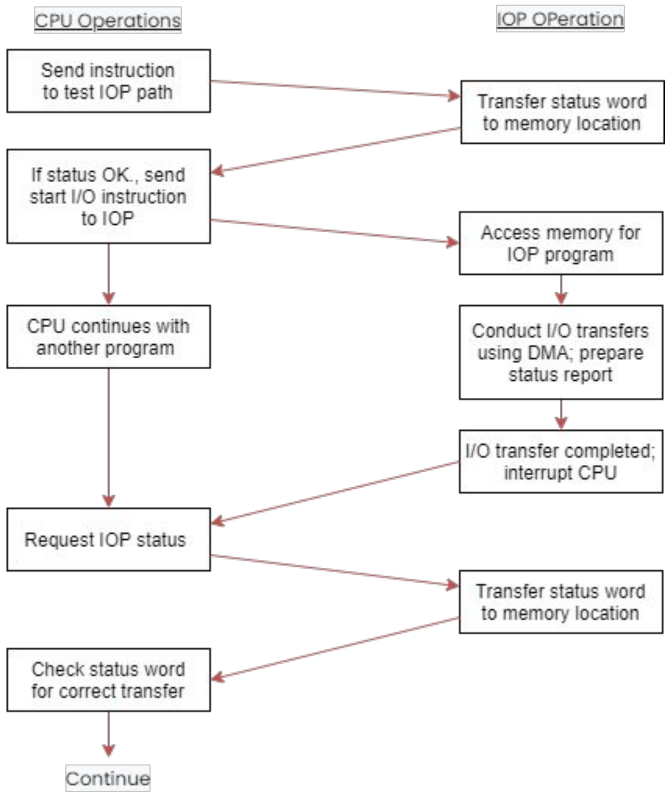
Input Output Processor

- The computer system with IOP can be divided into 3 separate modules:
  1. The Memory unit
  2. The CPU and
  3. one or more IOPS.
- The IOP can fetch and execute its own instructions.
- IOP instructions are specifically designed to facilitate I/O transfers.
- The IOP can also perform other processing tasks, such as arithmetic, logic, branching, and code translation.



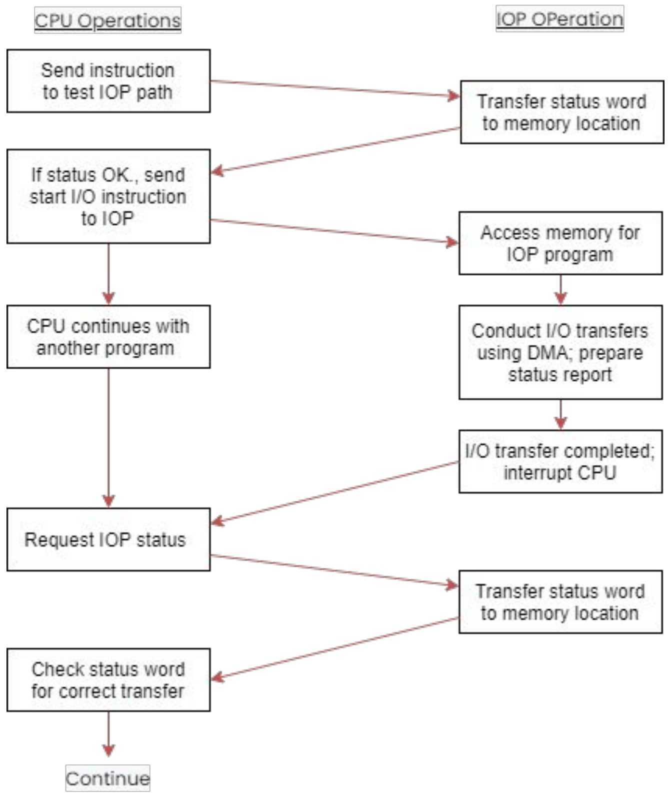
CPU-IOP Communication

- Instructions that are read from memory by an IOP are sometimes called commands, to distinguish them from instructions that are read by the CPU.
- The CPU sends an instruction to test the IOP path.
- The IOP responds by inserting a status word in memory for the CPU to check.
- The bits of the status word indicate the condition of the IOP and I/O device (IOP overload condition, device busy with another transfer, or device ready for I/O transfer).
- The CPU refers to the status word in memory to decide what to do next.
- If all is in order, the CPU sends the instruction to start I/O transfer.
- The memory address received with this instruction tells the IOP where to find program.


CPU Operations
IOP OPeration
Send instruction to test IOP path
Transfer status word to memory location
If status OK., send start I/O instruction to IOP
Access memory for IOP program
CPU continues with another program
Conduct I/O transfers using DMA; prepare status report
I/O transfer completed; interrupt CPU
Request IOP status
Transfer status word to memory location
Check status word for correct transfer
Continue

CPU-IOP Communication

- The CPU can now continue with another program while the IOP is busy with the I/O program.
- Both programs refer to memory by means of DMA transfer.
- When the IOP terminates the execution of its program, it sends an interrupt request to the CPU.
- The CPU responds to the interrupt by issuing an instruction to read the status from the IOP.
- The IOP responds by placing the contents of its status report into a specified memory location.
- The status word indicates whether the transfer has been completed or if any errors occurred during the transfer.
- From inspection of the bits in the status word, the CPU determines if the I/O operation was completed satisfactorily without errors.



CPU Operations
IOP OPeration
Send instruction to test IOP path
Transfer status word to memory location
If status OK., send start I/O instruction to IOP
Access memory for IOP program
CPU continues with another program
Conduct I/O transfers using DMA; prepare status report
I/O transfer completed; interrupt CPU
Request IOP status
Transfer status word to memory location
Check status word for correct transfer
Continue

Study at Home

Notes
Learn topics better with a little less effort utilizing notes by subject experts.

Syllabus
View or download syllabus for Bsc Computer Science under Calicut University.

Video Lessons
Better learning experience through video lectures from skilled personnel.

Question Papers
Familiarize with exam patterns through our library of previous year question papers.

Quiz
Test your knowledge of topics through various quizzes.

Solved Questions
Vast collection of questions and answers related to each topic.

Programming Laboratory
Practicals made easy with complete list of tested programs.

Software Downloads
Practice programming by downloading and installing respective softwares.

Teaching Resources
Find PPTs and related resources for teaching aid here.

teachics.

The Computer Science Learning Platform.