UNIT-IV

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**.**

Peripheral Devices:

- The input-output subsystem of a computer, referred to as I/O, provides an efficient mode of communication between the central system and the outside environment.
- Programs and data must be entered into computer memory for processing and results obtained from computations must be recorded or displayed for the user.
- The most common input output devices are:
  - ✓ Monitor
  - ✓ Keyboard
  - ✓ Mouse
  - ✓ Printer
  - ✓ Magnetic tapes

The devices that are under the direct control of the computer are said to be connected online.

- The input-output organization of a computer is a function of the size of the computer and the devices connected to it.
- The difference between a small and a large system is mostly dependent on the amount of hardware the computer has available for communicating with peripheral units and the number of peripherals connected to the system.

ASCII Alphanumeric Characters:

- Input and output devices that communicate with people and the computer are usually involved in the transfer of alphanumeric information to and from the device and the computer is ASCII
- (American Standard Code for Information Interchange).
- It uses seven bits to code 128 characters. The seven bits of the code are designated by b1 through b7 being the most significant bit.
- The ASCII code contains 94 characters that can be printed and 34 nonprinting characters used for various control functions.
- The printing characters consist of the 26 uppercase letters A through Z, the 26 lowercase letters, the 10 numerals 0 through 9, and 32 special printable characters such as %, *, and $.

- The 34 control characters are designated in the ASCII table with abbreviated names.
- The control characters are used for routing data and arranging the printed text into a prescribed format. There are three types of control characters: format effectors, information separators, and communication control characters.
- Format effectors are characters that control the layout of printing. They include the familiar typewriter controls, such as backspace (BS), horizontal tabulation (HT), and carriage return (CR).
- Information separators are used to separate the data into divisions like paragraphs and pages. They include characters such as record separator (RS) and file separator (FS).
- The communication control characters are useful during the transmission of text between remote terminals.
- Examples of communication control characters are **STX** (start of text) and **ETX** (end of text), which are used to frame a text message when transmitted through a communication medium.

INPUT-OUTPUT (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 central processing unit. 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 electro mechanical and electromagnetic devices and their manner of operation of the CPU and memory, which are electronic devices. Therefore, a conversion of signal values may be needed.
2. The data transfer rate of peripherals is usually slower than the transfer rate of CPU and consequently, a synchronization mechanism may be needed.
3. Data codes and formats in the peripherals differ from the word format in the CPU and memory.
4. The operating modes of peripherals are different from each other and must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

- **To Resolve these differences**, computer systems include special hardware components between the CPU and Peripherals to supervise and synchronizes all input and out transfers.
- These components are called **Interface Units** because they interface between the processor bus and the peripheral devices.

I/O bus and Interface Modules:

- A typical communication link between the processor and several peripherals is shown in Fig. 4-A.
- The I/O bus consists of data lines, address lines, and control lines.
- The magnetic disk, printer, and terminal are employed in practically any general-purpose computer. 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.
- Each peripheral has its own controller that operates the particular electromechanical device.

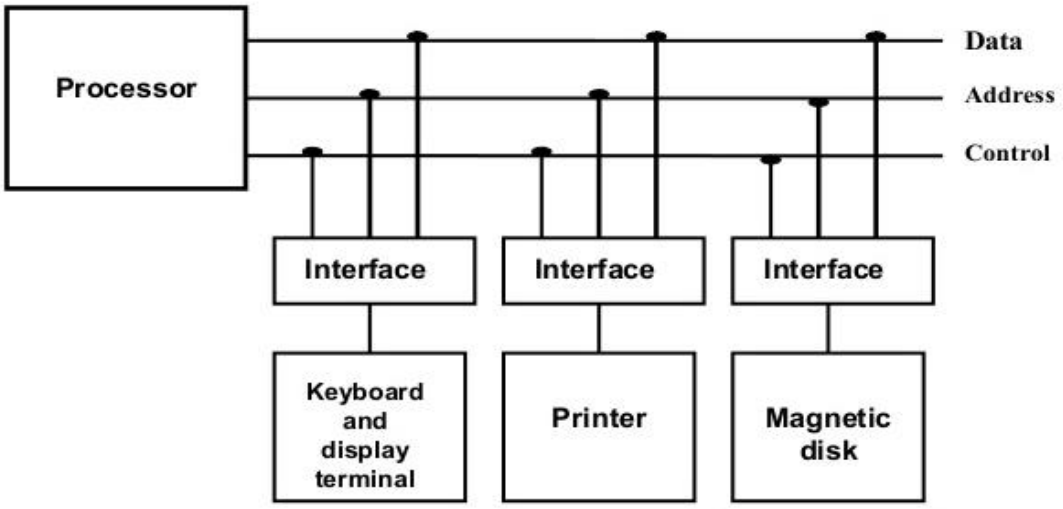


Figure 4-A: Connection of I/O bus to input devices.

- To communicate with a particular device, the processor places a device address on the address lines then processor provides a function code in the control lines. The function code is referred to as an I/O command.

The control lines are referred as an **I/O command**. The commands are as following:

1. **Control command-** A control command is issued to activate the peripheral and to inform it what to do.
2. **Status command-** A status command is used to test various status conditions in the interface and the peripheral.
3. **Output data command-** A data output command causes the interface to respond by transferring data from the bus into one of its registers.
4. **Input data command-** The data input command is the opposite of the data output. In this case the interface receives on item of data from the peripheral and places it in its buffer register.

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.
  2. Use one common bus for both memory and I/O but have separate control lines for each.
  3. Use one common bus for memory and I/O with common control lines.

Isolated Versus Memory-Mapped I/O

Isolated I/O:

- Many computers use one common bus to transfer information between memory or I/O and the CPU.
- In the isolated I/O configuration, the CPU has distinct input and output instructions, and each of these instructions is associated with the address of an interface register.
- When the CPU fetches and decodes the operation code of an input or output instruction, it places the address associated with the instruction into the common address lines.
- At the same time, it enables the I/O read (for input) or I/O write (for output) control line.
- This informs the external components that are attached to the common bus that the address in the address lines is for an interface register and **not for a memory word**.
- On the other hand, when the CPU is fetching an instruction or an operand from memory, it places the memory address on the address lines and enables the memory read or memory write control line.
- This informs the external components that the address is for a memory word and **not for an I/O** interface. The isolated I/O method isolates memory word and not for an I/O addresses.

Memory-Mapped I/O:

- In a memory-mapped I/O organization 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. Each interface is organized as a set of registers that respond to read and write requests in the normal address space.
- Computers with memory-mapped I/O can use memory-type instructions to access I/O data. It allows the computer to use the same instructions for either input-output transfers or for memory transfers.

ASYNCHRONOUS DATA TRANSFER

- The internal operations in a digital system are synchronized by means of clock pulses supplied by a common pulse generator.
- Two units, such as a CPU and an I/O interface, are designed independently of each other.
- If the registers in the interface share a common clock with the CPU registers, the transfer between the two units is said to be **synchronous**.
- In most cases, the internal timing in each unit is independent from the other in that each uses its own private clock for internal registers.
- In that case, the two units (CPU and an I/O interface) are said to be **asynchronous** to each other. This approach is widely used in most computer systems.
- Asynchronous data transfer between two independent units (CPU and an I/O interface) requires that control signals be transmitted between the communicating units to indicate the time at which data is being transmitted. There are two methods :
    a) Strobe Control
    b) Handshaking Method

a) Strobe Control:

The strobe control method of asynchronous data transfer employs a single control line to time each transfer. The strobe may be activated by either **the source or the destination unit**.

Source-initiated transfer:

- The data bus carries the binary information from source unit to the destination unit. Typically, the bus has multiple lines to transfer an entire byte or word.
- The strobe is a single line that informs the destination unit when a valid data word is available in the bus.

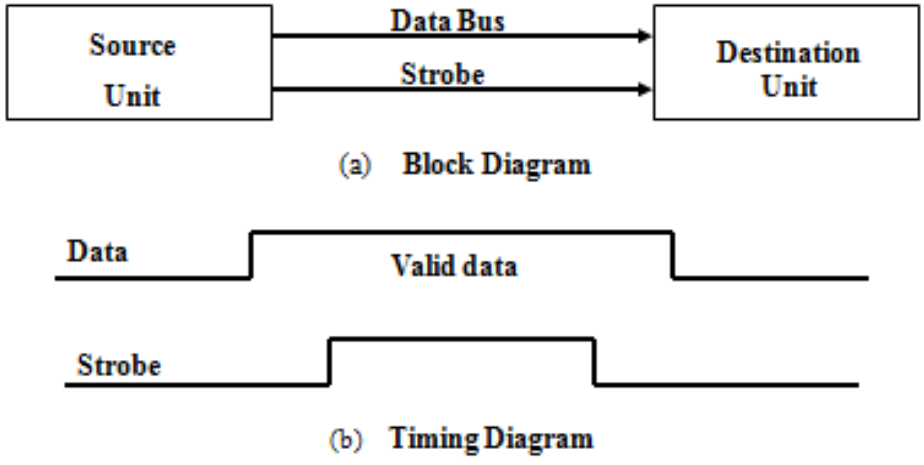


Fig 4-B: Source-initiated transfer a) Block-Diagram b) Timing diagram

- As shown in the timing diagram of Fig. (b), 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 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.
- Often, the destination unit uses the falling edge of the strobe pulse to transfer the contents of the data bus into one of its internal registers.
- The source removes the data from the bus a brief period after it disables its strobe pulse.

Destination-initiated transfer:

- In this case 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.

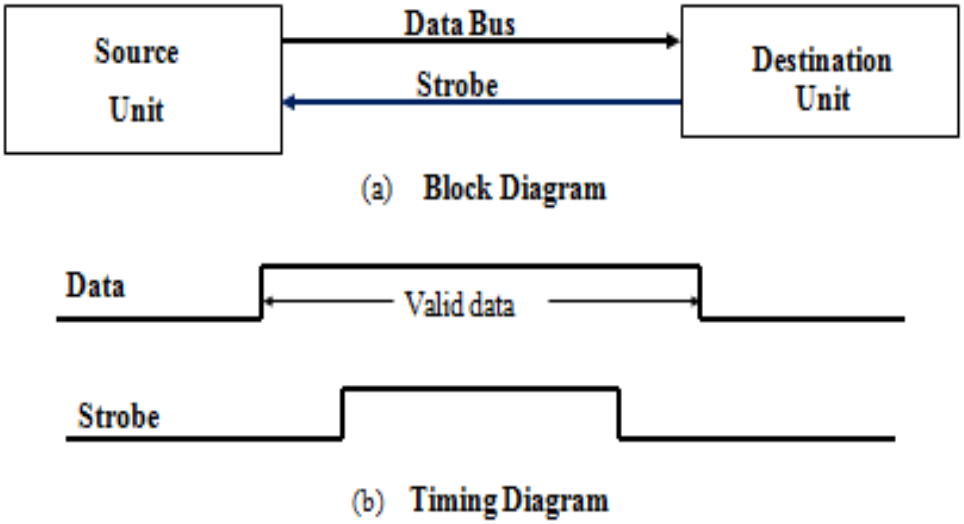


Fig 4-C: Destination-initiated transfer a) Block-Diagram b) Timing diagram

Disadvantage of Strobe Signal:

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

☞ The **Handshaking method** solves this problem.

b) Handshaking method:

- The handshake method introducing a second control signal that provides a reply to the unit that initiates the transfer.
- The basic principle of handshaking method of data transfer is as follows.
  - 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 valued 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.

Source-initiated transfer:

Figure 4-D: shows the data transfer initiated by the source

- 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

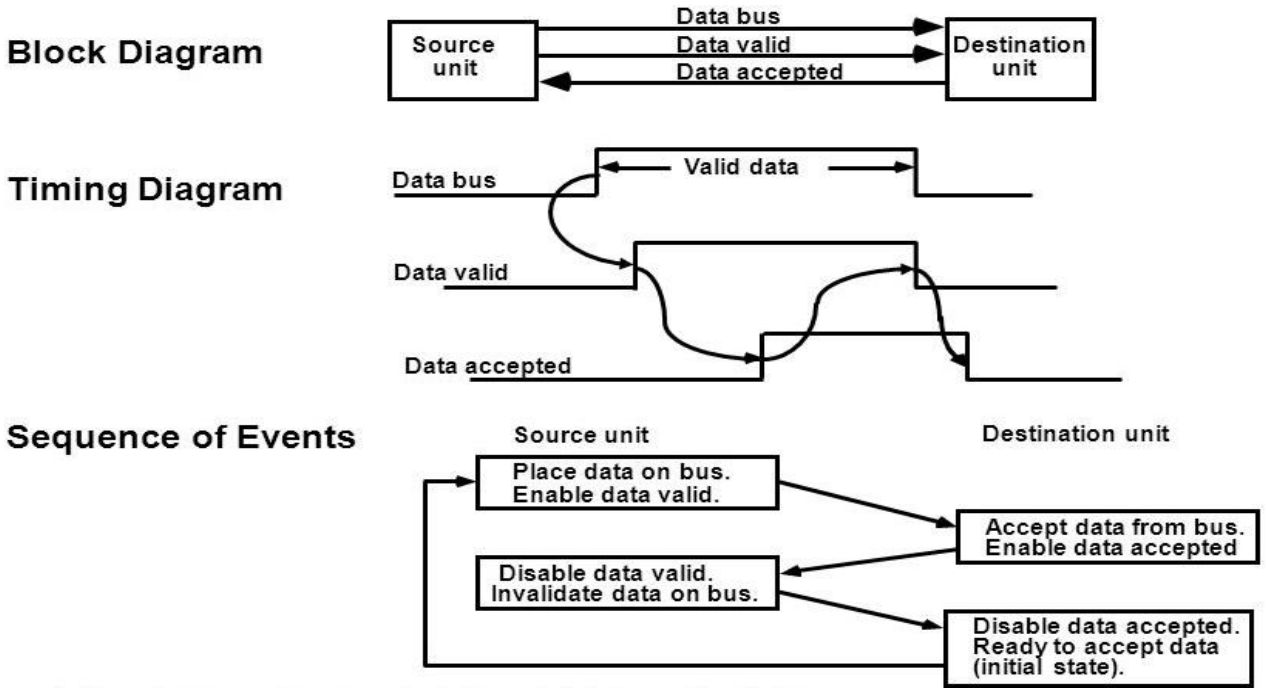


Figure 4-D: Source-initiated transfer using handshaking

Destination-initiated transfer:

Figure 4-E: shows the data transfer initiated by the Destination

- Note that the name of the signal generated by the destination unit has been changed to **ready for data** to reflect its new meaning.
- 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.
- Note that the sequence of events in both cases would be identical if we consider the ready for data signal as the complement of data accepted.
- In fact, the only difference between the source-initiated and the destination initiated transfer is in their choice of initial state.

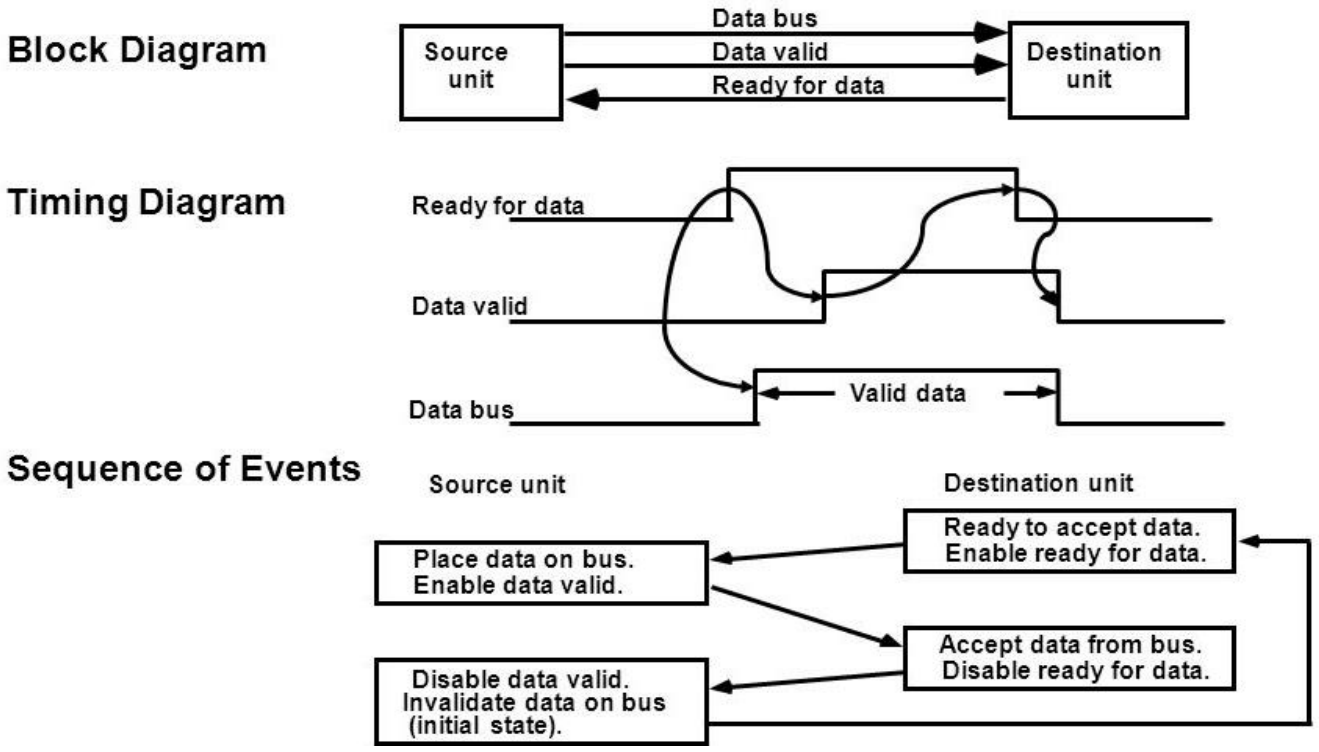


Figure 4-E: Destination-initiated transfer using handshaking

MODES OF TRANSFER

- Data transfer between the central computer and I/O devices may be handled in a variety of modes.
- Some modes use the CPU as an intermediate path; other transfers the data directly to and from the memory unit.
- Data transfer to and from peripherals may be handled in one of three possible modes:

1. Programmed I/O (or) Programmed Driven Method
2. Interrupt-initiated I/O (or) Interrupt Driven Method
3. Direct memory access (DMA)

1. Programmed I/O (or) Programmed Driven Method:

- In this mode of data transfer the operations are the results in **I/O instructions** which is a part of computer program. .

Example of Programmed I/O:

The data transfer from an I/O device through an interface into the CPU is shown in Fig. 4-F.

- When a byte of data is available, the device places it in the I/O bus and enables its data valid line.
- The interface accepts the byte into its data register and enables the data accepted line.
- The interface sets a bit in the status register that we will refer to as an F or "flag" bit. If the flag is equal to 1, the CPU reads the data from the data register.
- The device can now disable the data valid line, but it will not transfer another byte until the data accepted line is disabled by the interface.

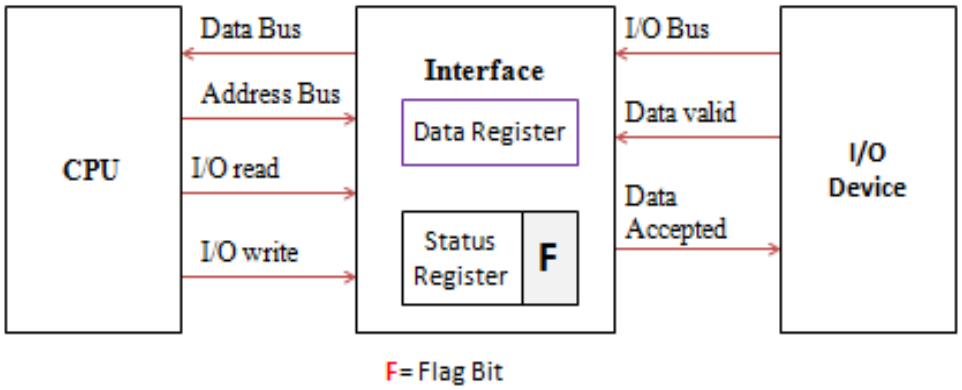


Figure 4-F: Data transfer form I/O device to CPU

- A flowchart of the program that must be written for the CPU is shown in Fig. 4-G. It is assumed that the device is sending a sequence of bytes that must be stored in memory.
- The transfer of each byte requires three instructions:

- ✓ Read the status register.
- ✓ Check the status of the flag bit and branch to step 1 if not set or to step 3 if set.
- ✓ Read the data register

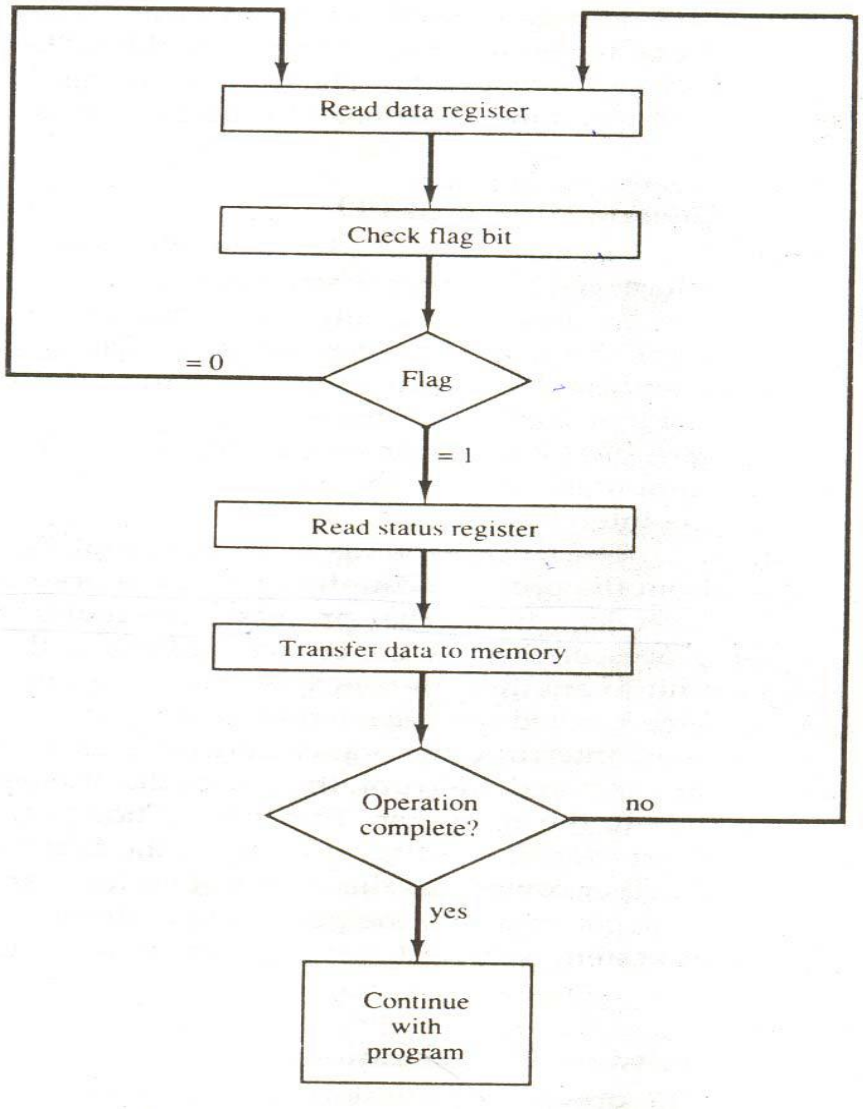


Figure 4-G: Flowchart for CPU program to input data.

Drawback of the Programmed I/O

- Each data transfer is initiated by a instruction in the program. Once the data is initiated the CPU starts monitoring the interface to see when next transfer can made.
- Thus 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 and the CPU time is wasted a lot in the executing of program.

☞ To remove this problem an Interrupt facility and special commands are used.

2. Interrupt-initiated I/O (or) Interrupt Driven Method:

- 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. When the CPU receives such an signal, it temporarily stops the execution of the program and branches to a service program to process the I/O transfer and.
- After completing it returns back to task, what it was originally performing.

Priority Interrupt:

- A priority interrupt is a system that determines which condition is to be serviced first when two or more requests arrive simultaneously.
- Highest priority interrupts are serviced first. Devices with high speed transfers 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.
- Establishing the priority of simultaneous interrupts can be done by **software or hardware**.

Daisy-chaining Priority:

- The daisy-chaining method of establishing priority consists of a serial connection of all devices that request an interrupt.
- The hardware priority function can be established by either a serial or a parallel connection of interrupt lines. **The serial connection is called daisy chaining method.**
- In daisy chaining method all the devices are connected in serial. The device with the **highest priority is placed in the first position**, followed by lower priority devices
- 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.
- The interrupt line stays in the high-level state and no interrupts are recognized by the CPU, only when no interrupts are pending.
- 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 the next device through the PO(Priority Out) Output only if device 1 is not requesting an interrupt.

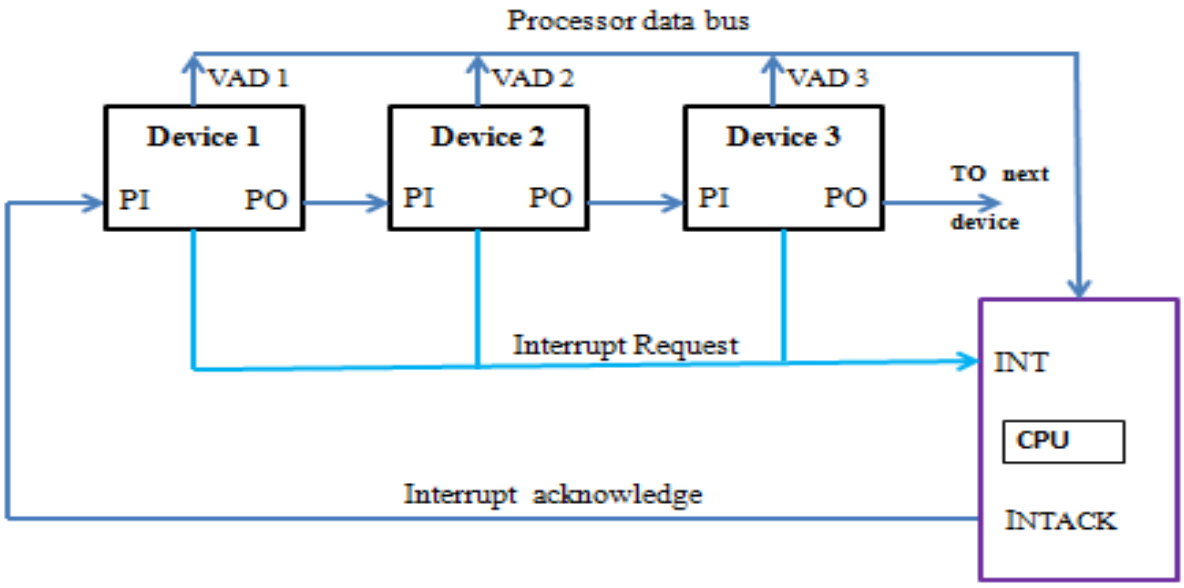


Figure 4-H: Daisy-chain priority interrupt

- It blocks the acknowledgement signal from the next device by placing a 0 in the PO output, if device I has a pending interrupt.
- 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 generate 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.
- It transmits the acknowledge signal to the next device by placing in 1 in its PO Output, if the device does not have pending interrupts.
- Thus the device with PI = 1 and PO = 0 is the one with the highest priority that is requesting an interrupt, and this device places its vector address (VAD) on the data.
- The daisy chain arrangement gives the highest priority to the device that receives the interrupt acknowledge signal from the CPU. The farther the device is from the first position; the lower is its priority.

3. Direct memory access (DMA)

- In the Direct Memory Access (DMA) the interface transfer the data into and out of the memory unit through the memory bus.
- The transfer of data between a fast storage device such as magnetic disk and memory is often limited by the speed of the CPU.
- **Removing the CPU from the path** and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This transfer technique is called **Direct Memory Access (DMA).**

- The CPU may be placed in an idle state in a variety of ways. One common method extensively used in microprocessors is to disable the buses through special control signals. Figure 4-15 shows two control signals in the CPU that facilitate the DMA transfer.

- **Bus Request (BR):** The Bus Request (BR) input is used by the DMA controller to request the CPU. When this input is active, the CPU terminates the execution of the current instruction and places the address bus, data bus and read write lines into a high Impedance state. High Impedance state means that the output is disconnected.
- **Bus Grant (BG)*:*** The CPU activates the ***Bus Grant (BG)*** output to inform the external DMA that the Bus Request (BR) can now take control of the buses to conduct memory transfer without processor.

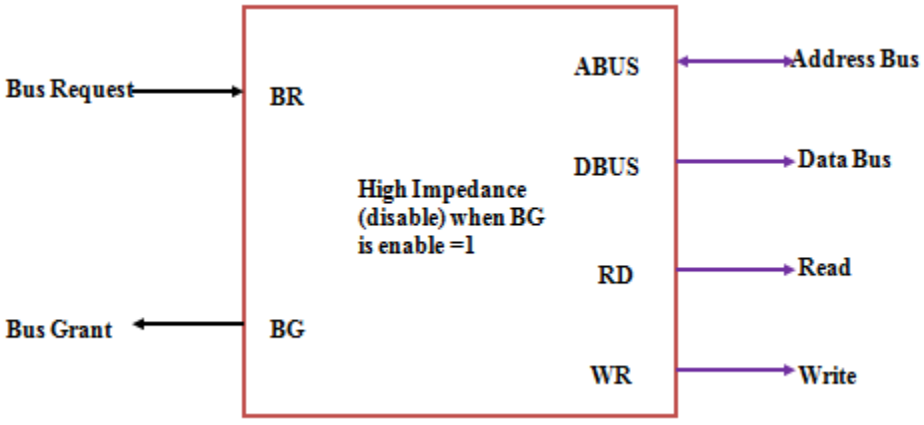


Figure 4-I: CPU bus signals for DMA transfer.

- When the DMA terminates the transfer, it disables the ***Bus Request (BR)*** line. The CPU disables the ***Bus Grant (BG)***, takes control of the buses and return to its normal operation

- The transfer can be made in several ways that are:

i) **DMA Burst: -** In DMA Burst transfer, a block sequence consisting of a number of memory words is transferred in continuous burst while the DMA controller is master of the memory buses.

ii) **Cycle Stealing** :- Cycle stealing allows the DMA controller to transfer one data word at a time, after which it must returns control of the buses to the CPU

DMA Controller:

Figure 4-J shows the block diagram of a typical DMA controller.

- The unit communicates with the CPU via the data bus and control lines. The registers in the DMA are selected by the CPU through the address bus by enabling the DS (DMA select) and RS (register select) inputs.
- The RD (read) and WR (write) inputs are bidirectional.
- When the BG (bus grant) input = 0, the CPU can communicate with the DMA registers through the data bus to read from or write to the DMA registers.
- When BG = 1, the CPU has relinquished the buses and the DMA can communicate directly with the memory by specifying an address in the address bus and activating the RD or WR control.

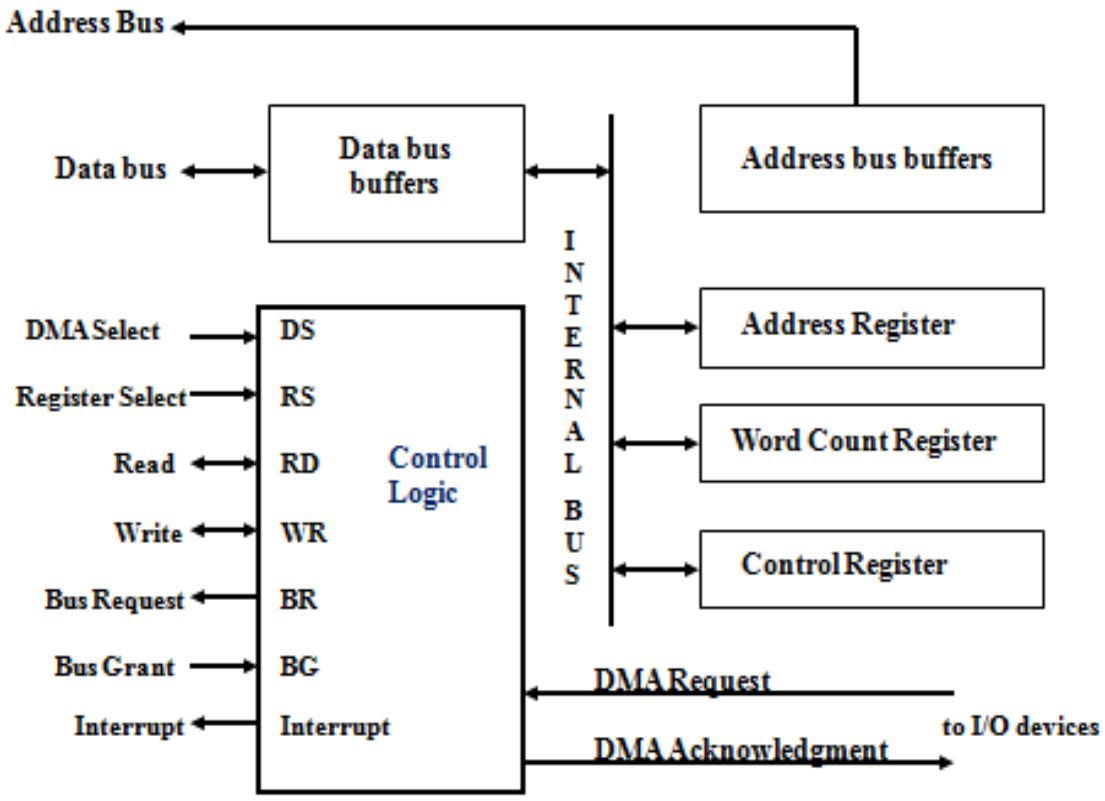


Figure 4-J: Block diagram of DMA controller.

- The DMA controller has three registers:
  i. **Address Register: -** Address Register contains an address to specify the desired location in memory.
  ii. **Word Count Register: -** WC holds the number of words to be transferred. The register is increment/decrement by one after each word transfer and internally tested for zero.
  iii. **Control Register: -** Control Register specifies the mode of transfer.

DMA Transfer:

The position of the DMA controller among the other components in a computer system is illustrated in Fig. 4-K.

- The CPU communicates with the DMA through the address and data buses as with any interface unit.
- The DMA has its own address, which activates the DS and RS lines. The CPU initializes the DMA through the data bus.
- Once the DMA receives the start control command, it can transfer between the peripheral and the memory.

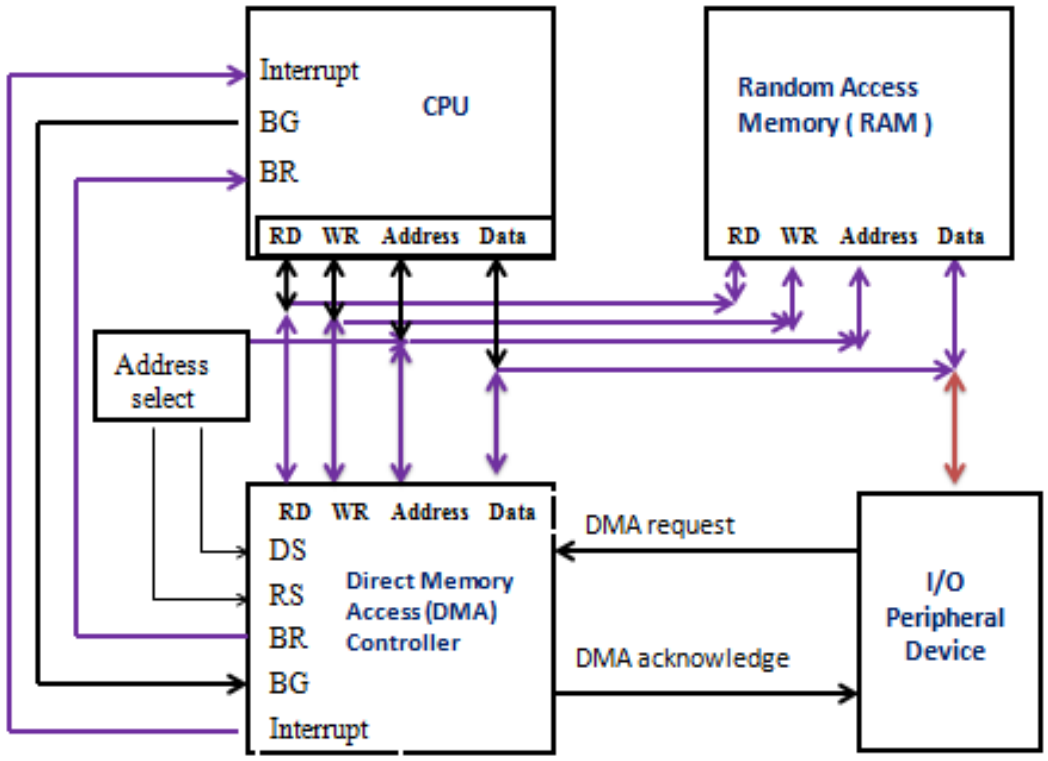


Figure 4-K: DMA transfer in a computer system.

- When the peripheral device sends a DMA request, the DMA controller activates the BR line, informing the CPU to relinquish the buses.

- The CPU responds with its BG line, informing the DMA that its buses are disabled. The DMA then 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 peripheral device.
- Note that the RD and WR lines in the DMA controller are bidirectional.
- When **BG = 0** the RD and WR are input lines allowing the CPU to communicate with the internal DMA registers.
- When **BG=1**, the RD and WR are output lines from the DMA controller to the random access memory to specify the read or write operation of data.

- DMA transfer is very useful in many applications. It is used for fast transfer of information between magnetic disks and memory.

Memory Organization

Memory Hierarchy:

- The total memory capacity of a computer can be visualized as being a hierarchy of components.
- Figure 4-L A illustrates the components in a typical memory hierarchy. At the bottom of the hierarchy are the relatively slow magnetic tapes used to store removable files. Next are the magnetic disks used as backup storage.
- The main memory occupies a central position by being able to communicate directly with the CPU and with auxiliary memory devices through an I/O processor.
- When programs not residing in main memory are 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 for currently used programs and data.
- Special very-high speed memory called a cache is sometimes used to increase the speed of processing by making current programs and data available to the CPU at a rapid rate.
- While the I/O processor manages data transfers between auxiliary memory and main memory, the cache organization is concerned with the transfer of information between main memory and CPU. Thus each is involved with a different level in the memory hierarchy system.

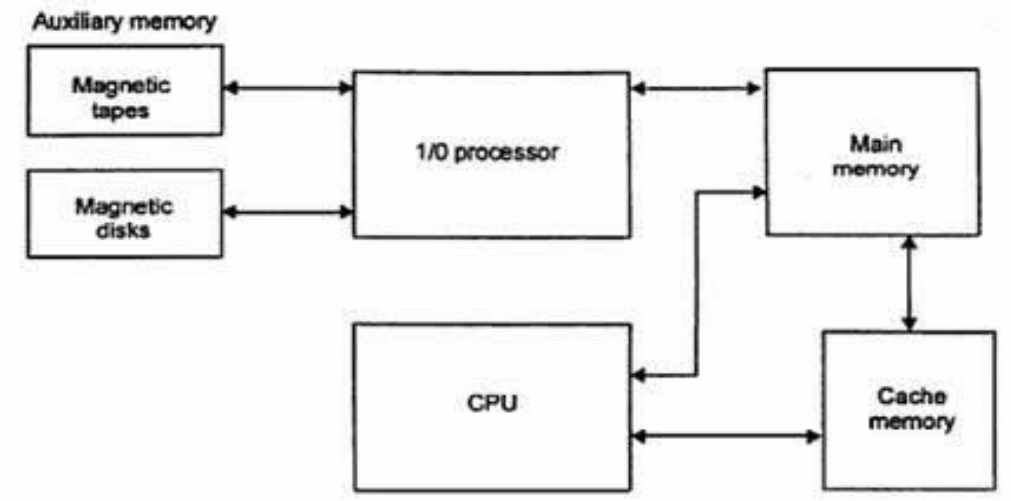


Figure 4-L: Memory hierarchy in a computer system.

- The reason for having two or three levels of memory hierarchy is economics. As the storage capacity of the memory increases, the cost per bit for storing binary information decreases and the access time of the memory becomes longer.
- The auxiliary memory has a large storage capacity, is relatively inexpensive, but has low access speed compared to main memory.
- The cache memory is very small, relatively expensive, and has very high access speed. Thus as the memory access speed increases, so does its relative cost.
- The overall goal of using a memory hierarchy is to obtain the highest-possible average access speed while minimizing the total cost of the entire memory system.
- **Auxiliary and cache memories are used for different purposes**. The cache holds those parts of the program and data that are most heavily used, while the auxiliary memory holds those parts that are not presently used by the CPU.

Main Memory:

- The main memory is the central storage unit in a computer system. It is a relatively large and fast memory used to store programs and data during the computer operation.
- The principal technology used for the main memory is based on semiconductor integrated circuits. Integrated circuit RAM chips are available in two possible operating modes, static and dynamic.
- 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
- RAM is used for storing the bulk of the programs and data that are subject to change. ROM is used for storing programs that are permanently resident in the computer and for tables of constants that do not change in value one the production of the computer is completed.
- Among other things, the ROM portion of main memory is needed for storing an initial program called a **bootstrap loader**. The bootstrap loader is a program whose function is to start the computer software operating when power is turned on.
- Since **RAM is volatile**, its contents are destroyed when power is turned off. The contents of ROM remain unchanged after power is turned off and on again.

Memory Address Map:

- The designer of a computer system must calculate the amount of memory required for the particular application and assign it to either RAM or ROM.

- The interconnection between memory and processor is then established from knowledge of the size of memory needed and the type of RAM and ROM chips available.
- The addressing of memory can be established 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.

TABLE 4-M: Memory Address Map for Microprocomputer

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

Example: Assume that a computer system needs 512 bytes of RAM and 512 bytes of ROM.

- The memory address map for this configuration is shown in Table 4-M. The component column specifies whether a RAM or a ROM chip is used.
- The hexadecimal address column assigns a range of hexadecimal equivalent addresses for each chip.
- The address bus lines are listed in the third column. Although there are 16 lines in the address bus, the table shows only 10 lines because the other 6 are not used in this example and are assumed to be zero.
- The small x's under the address bus lines designate those lines that must be connected to the address inputs in each chip.
- 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 are 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. Note that any other pair of unused bus lines can be chosen for this purpose.
- The distinction between a RAM and ROM address is done with another bus line. Here we choose line 10 for this purpose. When line 10 is 0, the CPU selects a RAM, and when this line is equal to 1, it selects the ROM.

Memory Connection to CPU:

- RAM and ROM chips are connected to a CPU through the data and address buses. The low-order lines in the address bus select the byte within the chips and other lines in the address bus select a particular chip through its chip select inputs.
- The connection of memory chips to the CPU is shown in Fig. 4-N. This configuration gives a memory capacity of 512 bytes of RAM and 512 bytes of ROM. It implements the memory map of Table 4-M.
- Each RAM receives the seven low-order bits of the address bus to select one of 128 possible bytes. The particular RAM chip selected is determined from lines 8 and 9 in the address bus.
- This is done through a 2 x 4 decoder whose outputs go to the CS1 inputs in each RAM chip. Thus, when address lines 8 and 9 are equal to 00, the first RAM chip is selected. When 01, the second RAM chip is selected, and so on.
- The RD and WR outputs from the microprocessor are applied to the inputs of each RAM chip.

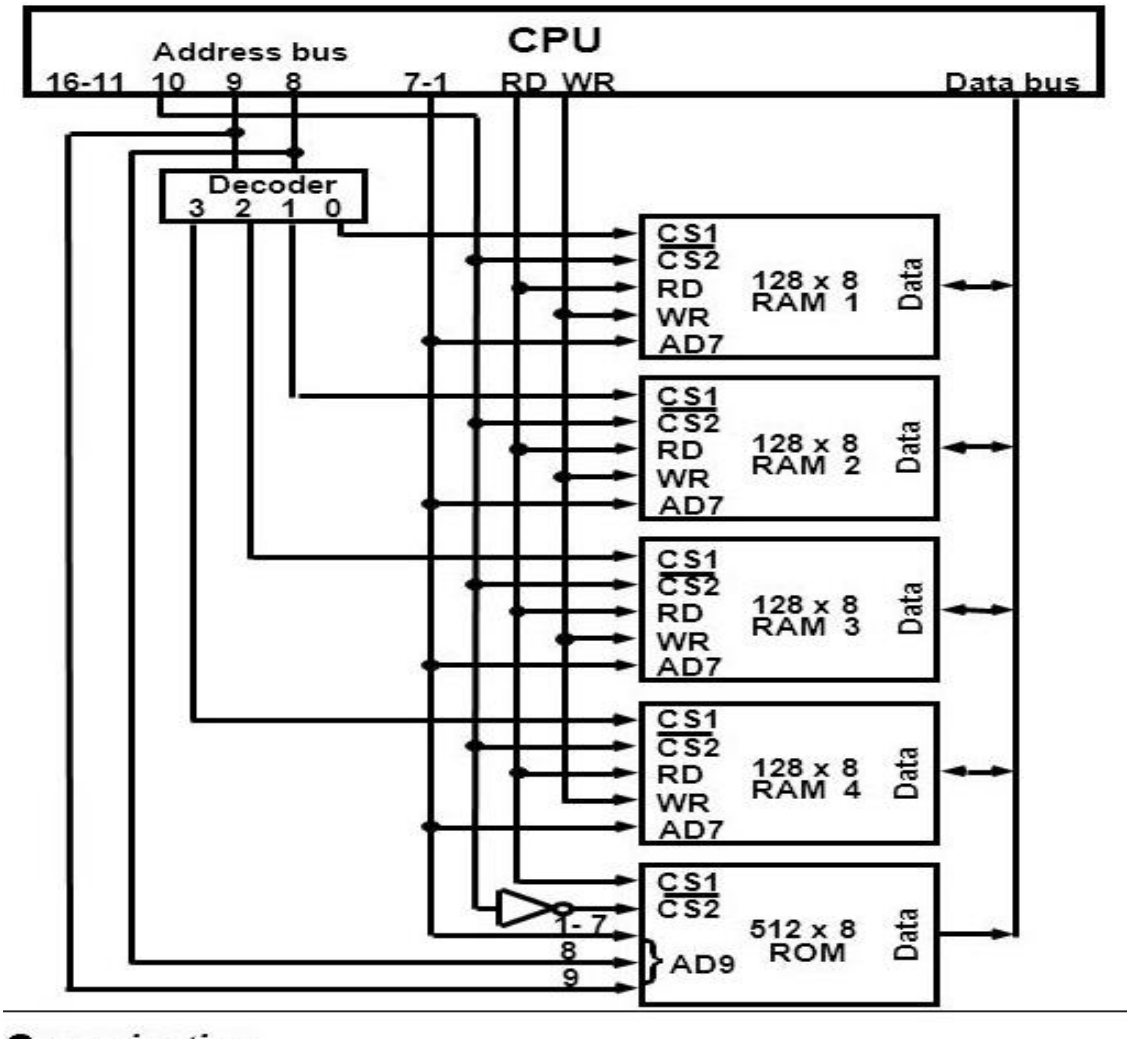


Fig. 4-N: Memory connection to the CPU.

- The selection between RAM and ROM is achieved through bus line 10. The RAMs are selected when the bit in this line is 0, and the ROM when the bit is 1.
- The other chip select input in the ROM is connected to the RD control line for the ROM chip to be enabled only during a read operation.
- Address bus lines 1 to 9 are applied to the input address of ROM without going through the decoder.
- The data bus of the ROM has only an output capability, whereas the data bus connected to the RAMs can transfer information in both directions.

Auxiliary Memory

- The important characteristics of any device are its access mode, access time, transfer rate, capacity, and cost.
- The average time required to reach a storage location in memory and obtain its contents is called the access time.
- In electromechanical devices with moving parts such as disks and tapes, the access time consists of a seek time required to position the read-write head to a location and a transfer time required to transfer data to or from the device

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).
- 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

Cache Memory:

- A **CPU cache** is a hardware cache used by the central processing unit (CPU) of a computer to reduce the average cost (time or energy) to access data from the main memory. A cache is a smaller, faster memory, closer to a processor core, which stores copies of the data from frequently used main memory locations.

Locality of reference:

- In computer science, locality of reference, also called the principle of locality, is the term applied to situations where the same value or related storage locations are frequently accessed. There are three basic types of locality of reference: temporal, spatial and sequential:
  - Temporal locality

    Here a resource that is referenced at one point in time is referenced again soon afterwards.
  - Spatial locality

    Here the likelihood of referencing a storage location is greater if a storage location near it has been recently referenced.
  - Sequential locality

    Here storage is accessed sequentially, in descending or ascending order.
- The basic characteristic of cache memory is its fast access time. Therefore, very little or no time must be wasted when searching for words in the cache.
- The **transformation of data from main memory to cache memory is referred to as a mapping process.** Three types of mapping procedures are of practical interest when considering the organization of cache memory:

  1. Associative mapping
  2. Direct mapping
  3. Set-associative mapping

Consider below diagram to discuss above three mapping techniques:

- 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. For every word stored in cache, there is a duplicate copy in main memory. The CPU communicates with both memories.
- It first sends a 15-bit address to cache. If there is a hit, the CPU accepts the 12-bit data from cache. If there is a miss, the CPU reads the word from main memory and the word is then transferred to cache.

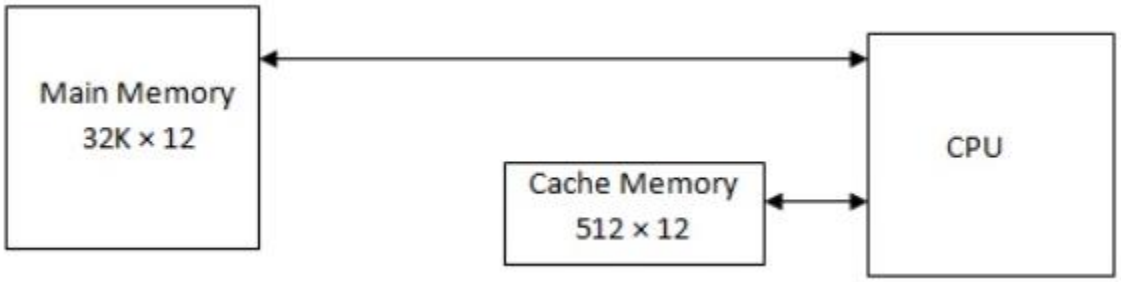


Figure 4-O: Example of cache memory.

1. **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.

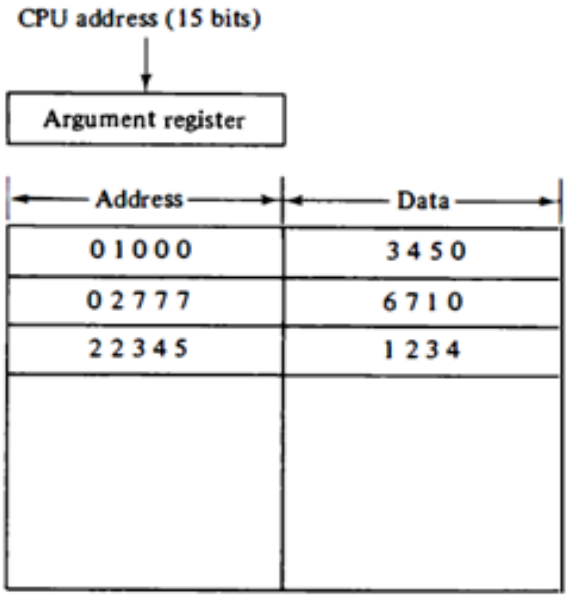


Figure 4-P: Associative mapping cache (all numbers in octal).

- 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.
- The decision as to what pair is replaced is determined from the replacement algorithm that the designer chooses for the cache.

❖ **Disadvantage of associate mapping**:

- Associative memories are expensive compared to random-access memories because of the added logic associated with each cell.

2. Direct Mapping:

- Here 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 figure shows that 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.

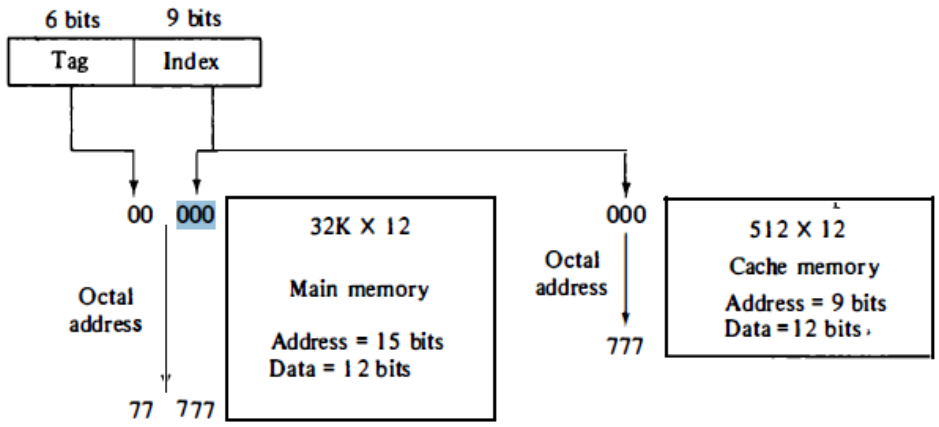


Figure 4-Q: Addressing relationships between main and cache memories.

- The internal organization of the words in the cache memory is as shown in Fig. 4-R(b).
- 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.

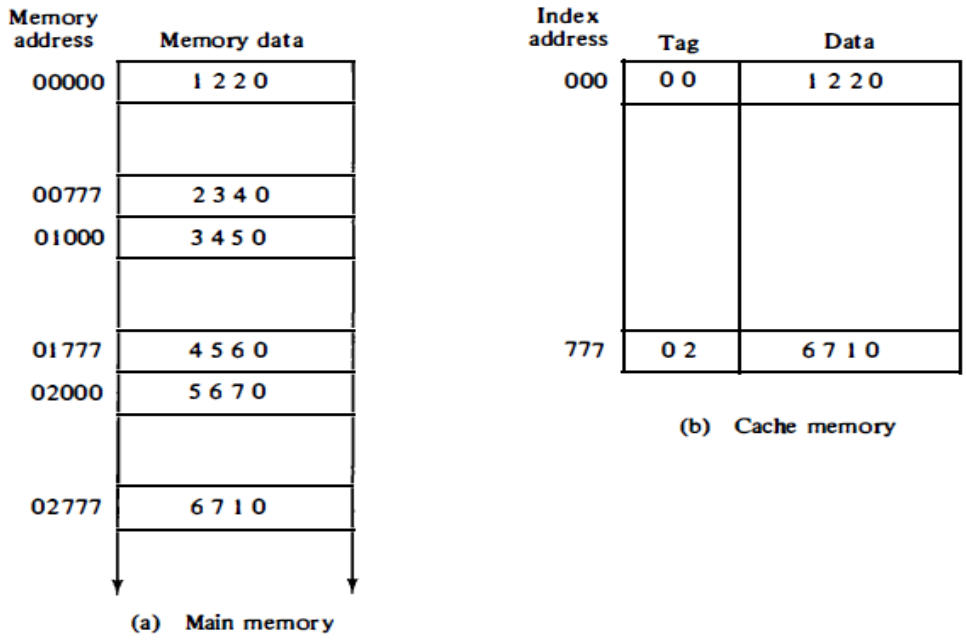


Figure 4-R: Direct mapping cache organization.

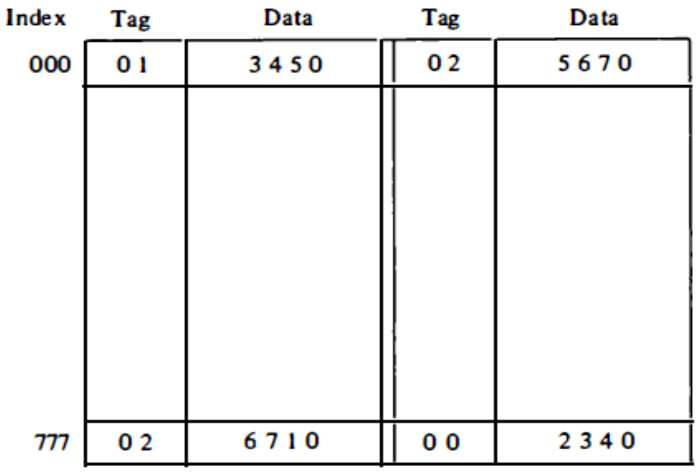
- The tag field of the CPU address is compared with the tag in the word read from the cache.
- If the two tags match, there is a hit and the desired data word is in cache. If there is no match, there is a miss and the required word is read from main memory.
- It is then stored in the cache together with the new tag, replacing the previous value.

❖ **The disadvantage of direct mapping**:

- The hit ratio can drop considerably if two or more words whose addresses have the same index but different tags are accessed repeatedly

3. Set-Associative Mapping:

- A third type of cache organization, called 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 Fig. 4-S.
- Each index address refers to two data words and their associated tags. Each tag requires six bits and each data word has 12 bits, so the word length is 2(6 + 12) = 36 bits. An index address of nine bits can accommodate 512 words. Thus the size of cache memory is 512 X 36.



Index	Tag	Data	Tag	Data
000	01	3450	02	5670
777	02	6710	00	2340

Figure 4-S: Two-way set-associative mapping cache.

- The words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000. Similarly, the words at addresses 02777 and 00777 are stored in cache at index address 777.
- 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."

❖ **Advantage of set-associate mapping:**

- The hit ratio will improve as the set size increases because more words with the same index but different tags can reside in cache.

Writing into Cache:

- ❖ An important aspect of cache organization is concerned with memory write requests. When the CPU finds a word in cache during a read operation, the main memory is not involved in the transfer.
- ❖ However, if the operation is a write, there are two ways that the system can precede.

Write-through method:

- 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. This method has the advantage that main memory always contains the same data as the cache. This characteristic is important in systems with direct memory access transfers (DMA).

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.
- The reason for the write-back method is that during the time a word resides in the cache, it may be updated several times; however, as long as the word remains in the cache, it does not matter whether the copy in main memory is out of date, since requests from the word are filled from the cache.
- It is only when the word is displaced from the cache that an accurate copy need be rewritten into main memory**.**