FCO (CC-201)

Unit-4

Organization of Input-Output and Memory

Input-Output Organization

What is Peripheral device? Explain its types in detail.

- **Peripheral device**, also known as **computer peripheral,** or **input/output device**, any of various devices (including sensors) used to enter information and instructions into a computer for storage or processing and to deliver the processed data to a human operator.
- Peripheral are commonly divided into three kinds**:**

  1. Input devices,

  2. Output devices, and

  3. Storage devices.
- An **input device** converts incoming data and instructions into a pattern of electrical signals in binary code that are understandable to a digital computer.
- Input devices include typewriter-like keyboards; handheld devices such as the mouse trackball, joystick, trackpad, and special pen with pressure-sensitive pad; microphones, webcams, and digital cameras.
- They also include sensors that provide information about their environment tempture , pressure and so forth—to a computer.
- Another direct-entry mechanism is the optical laser scanner (e.g., scanners used with point-of-sale terminals in retail stores) that can read bar-coded data or optical character fonts.

- An **output device** reverses the process, translating the digitized signals into a form understandable to the user.
- Output equipment includes video display terminal ink-jet and laser printers, loudspeakers, headphones, and devices such as flow valves that control machinery, often in response to computer processing of sensor input data.
- Some devices, such as video display terminals and USB hubs, may provide both input and output.
- Other examples are devices that enable the transmission and reception of data between computers, e.g., modems and network interfaces.

Input Devices

- Keyboard
- Optical input devices
  - Card Reader
  - Paper Tape Reader
  - Bar code reader
  - Digitizer
  - Optical Mark Reader
- Magnetic Input Devices
  - Magnetic Stripe Reader
- Screen Input Devices
  - Touch Screen
  - Light Pen
  - Mouse
- Analog Input Devices

Output Devices

- Card Puncher, Paper Tape Puncher
- CRT
- Printer (Impact, Ink Jet, Laser, Dot Matrix)
- Plotter
- Analog
- Voice

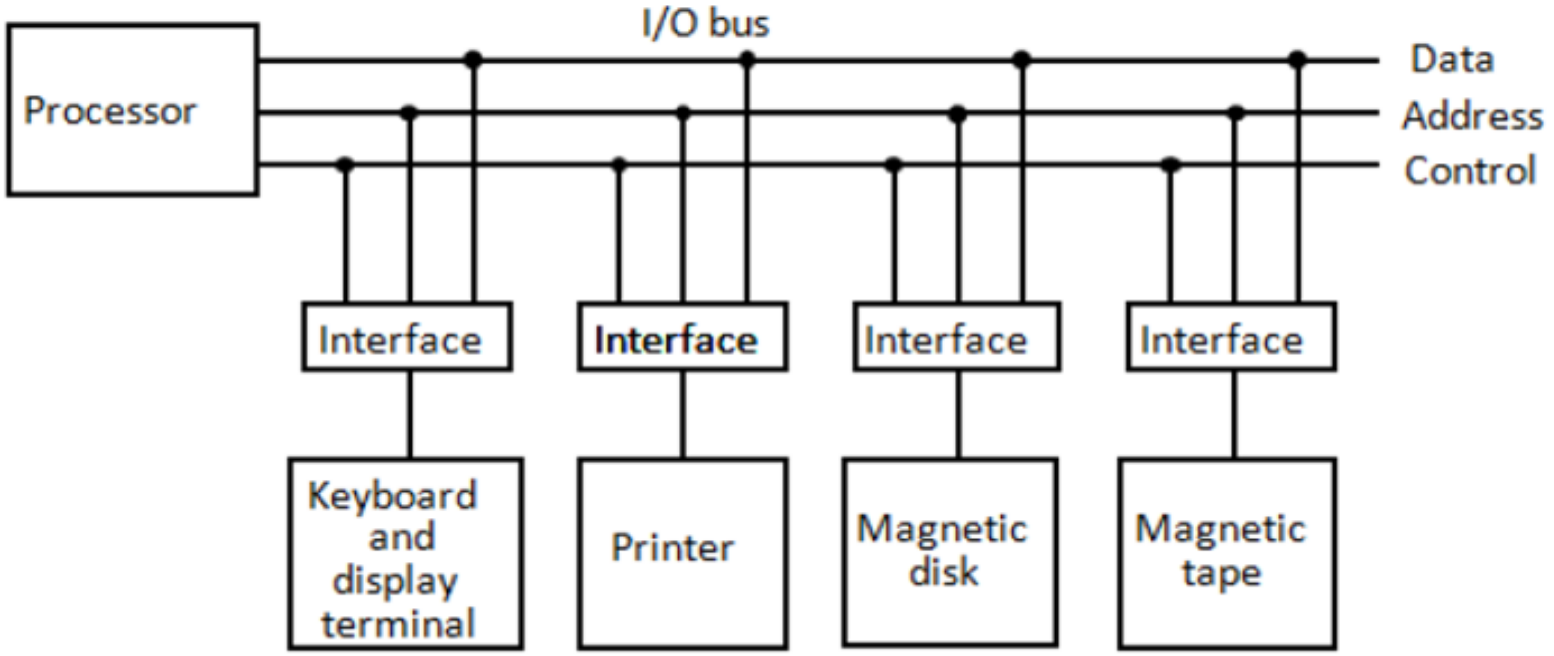
- Most auxiliary **storage devices**—as, for example, CD-ROM and DVD drives, flash memory drives, and external disk drives also double as input/output devices.
- Even devices such as smart phone, tablet computer and wearable devices like fitness trackers and smart watch can be considered as peripherals, even if ones that can function independently.

I/O Interface

Define Peripherals. Explain I/O Bus and Interface Modules.

Peripherals:

Input-output device attached to the computer are also called peripherals.



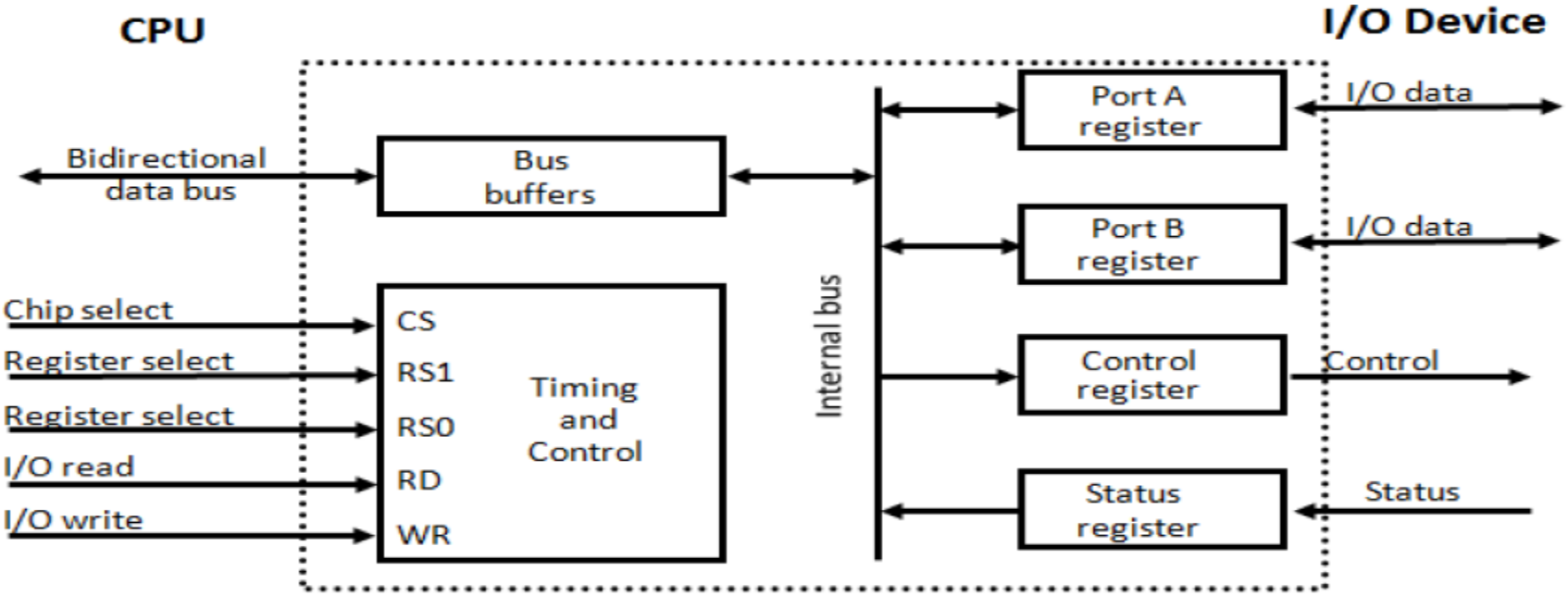
Connection of I/O bus to input-output device.

- A typical communication link between the processor and several peripherals is shown in figure
- 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.
- It also synchronizes the data flow and supervises the transfer between peripheral and processor.
- Each peripheral has its own controller that operates the particular electromechanical device.
- For example, the printer controller controls the paper motion, the print timing, and the selection of printing characters.
- The I/O bus from the processor is attached to all peripheral interfaces.
- 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 peripherals whose address does not correspond to the address in the bus are disabled by their interface selected responds to the function code and proceeds to execute it.

- The function code is referred to as an I/O command.
- There are four types of commands that an interface may receive. They are classified as control, status, data output, and data input.
- A control command is issued to activate the peripheral and to inform it what to do.
  For example, a magnetic tape unit may be instructed to backspace the tape by one record, to rewind the tape, or to start the tape moving in the forward direction.
- A status command is used to test various status conditions in the interface and the peripheral.
  For example, the computer may wish to check the status of the peripheral before a transfer is initiated.
- During the transfer, one or more errors may occur which are detected by the interface.
- These errors are designated by setting bits in a status register that the processor can read at certain intervals.
- A data output command causes the interface to respond by transferring data from the bus into one of its registers.
- The computer starts the tape moving by issuing a control command.
- The processor then monitors the status of the tape by means of a status command.
- When the tape is in the correct position, the processor issues a data output command.
- The interface responds to the address and command and transfers the information from the data lines in the bus to its buffer register.
- The interface that communicates with the tape controller and sends the data to be stored on tape.
- The data input command is the opposite of the data output.
- In this case 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.

Explain I/O interface with example.

- An example of an I/O interface units is shown in block diagram from in figure
- It consists of two data registers called ports, a control register, a status register, bus buffers, and timing and control circuit.
- The interface communicates with the CPU through the data bus.
- The chip select and register select inputs determine the address assigned to the interface.
- The I/O read and writes are two control lines that specify an input or output, respectively.
- The four registers communicate directly with the I/O device attached to the interface.
- The I/O data to and from the device can be transferred into either port A or port B.
- If the interface is connected to a printer, it will only output data, and if it services a character reader, it will only input data.
- A magnetic disk unit transfers data in both directions but not at the same time, so the interface can use bidirectional lines.
- A command is passed to the I/O device by sending a word to the appropriate interface register.
- The control register receives control information from the CPU. By loading appropriate bits into the control register, the interface and the I/O device attached to it can be placed in a variety of operating modes.



CS	RS1	RS0	Register Selected
0	X	X	None: data bus in high impedance
1	0	0	Port A register
1	0	1	Port B register
1	1	0	Control register
1	1	1	Status register

Figure 8.2: Example of I/O interface unit

- For example, port A may be defined as an input port and port B as an output port.
- A magnetic tape unit may be instructed to rewind the tape or to start the tape moving in the forward direction.
- The bits in the status register are used for status conditions and for recording errors that may occur during the data transfer.
  For example, a status bit may indicate that port A has received a new data item from the I/O device.
- Another bit in the status register may indicate that a parity error has occurred during the transfer.

- The interface registers communicate with the CPU through the bidirectional data bus.
- The address bus selects the interface unit through the chip select and the two register select inputs.
- A circuit must be provided externally (usually, a decoder) to detect the address assigned to the interface registers.
- This circuit enables the chip select (CS) input when the interface is selected by the address bus.
- The two register select inputs RS1 and RS0 are usually connected to the two least significant lines of the address bus.
- These two inputs select one of the four registers in the interface as specified in the table accompanying the diagram.
- The content of the selected register is transfer into the CPU via the data bus when the I/O read signal is enables.
- The CPU transfers binary information into the selected register via the data bus when the I/O write input is enabled.

Asynchronous Data Transfer

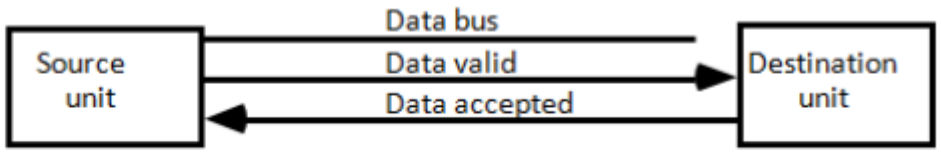
Explain Asynchronous data transfer with Handshaking method.

- The handshake method solves the problem of Strobe method by introducing a second control signal that provides a reply to the unit that initiates the transfer.

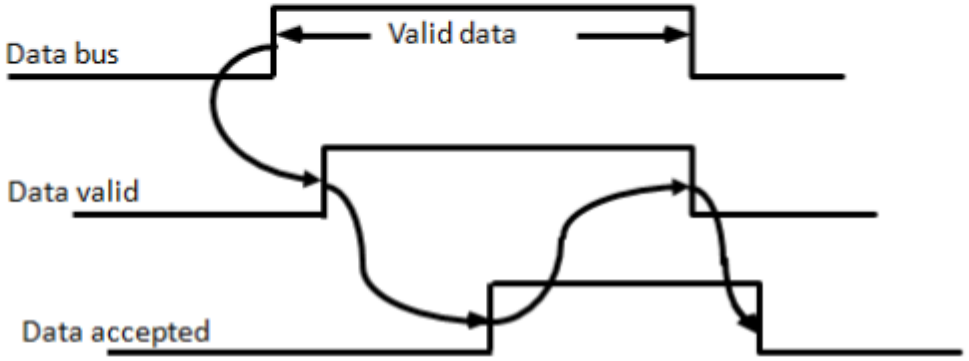
Source-initiated transfer using handshaking

- 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.
- Figure shows the data transfer procedure initiated by the source.
- The two handshaking lines the data valid, which is generated by the source unit, and data accepted, generated by the destination unit, the timing diagram shows the exchange of signals between the two units.
- The sequence of events listed in figure 8.5 shows the four possible states that the system can be at any given time.
- The source unit initiates the transfer by placing the data on the bus and enabling its data valid signal.

Block Diagram



Timing Diagram



Sequence of Events

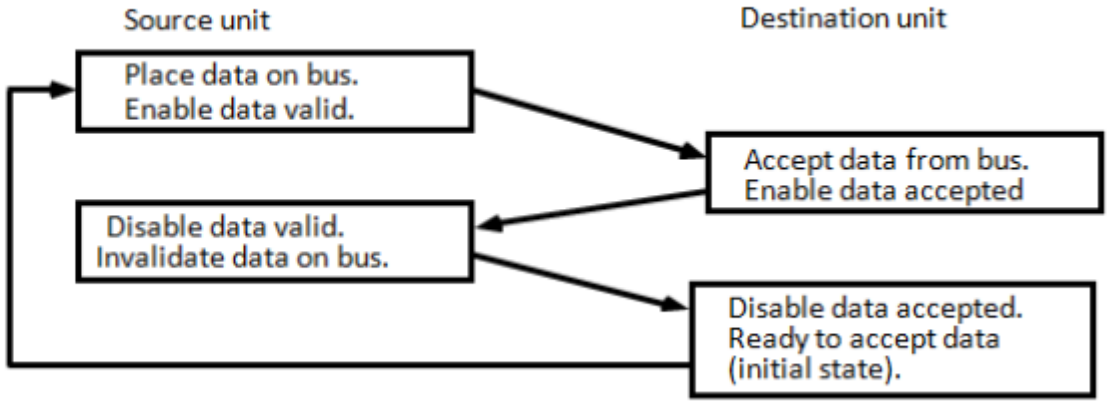


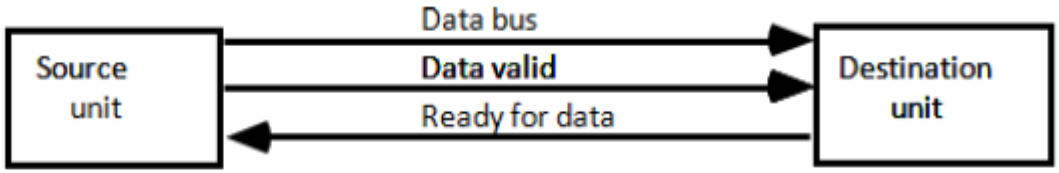
Figure 8.5: Source-initiated transfer using handshaking

- 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 arbitrary delays from one state to the next and permits each unit to respond at its own data transfer rate.

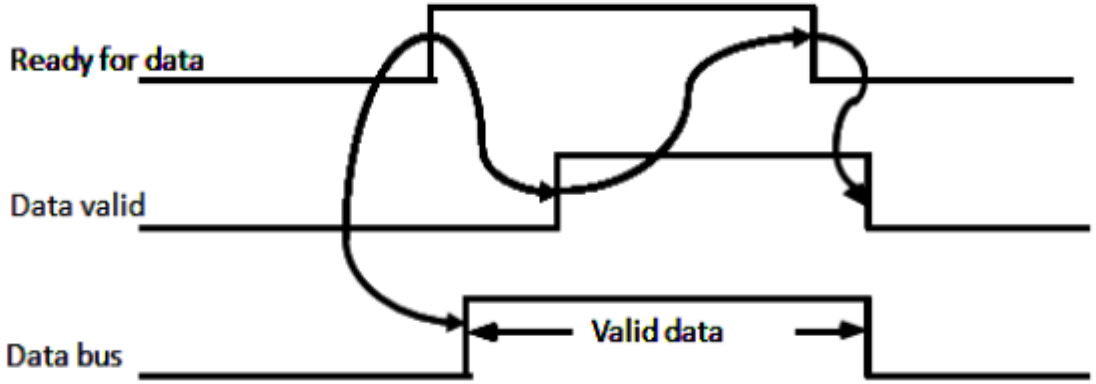
Destination-initiated transfer using handshaking

- The destination-initiated transfer using handshaking lines is shown in figure 8.6.
- 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.

Block Diagram



Timing Diagram



Sequence of Events

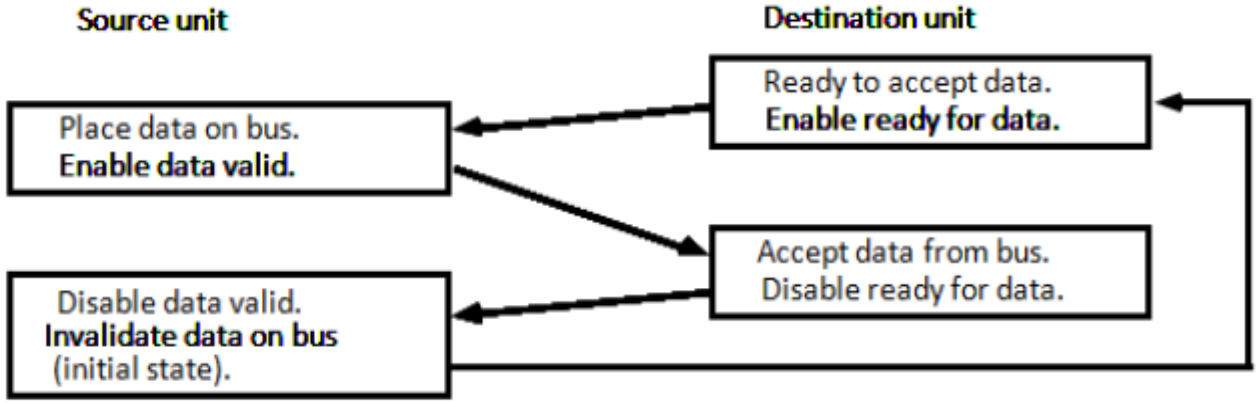


Figure 8.6: Destination-initiated transfer using handshaking

Modes of Transfer

Explain Programmed I/O with example.

Programmed I/O:

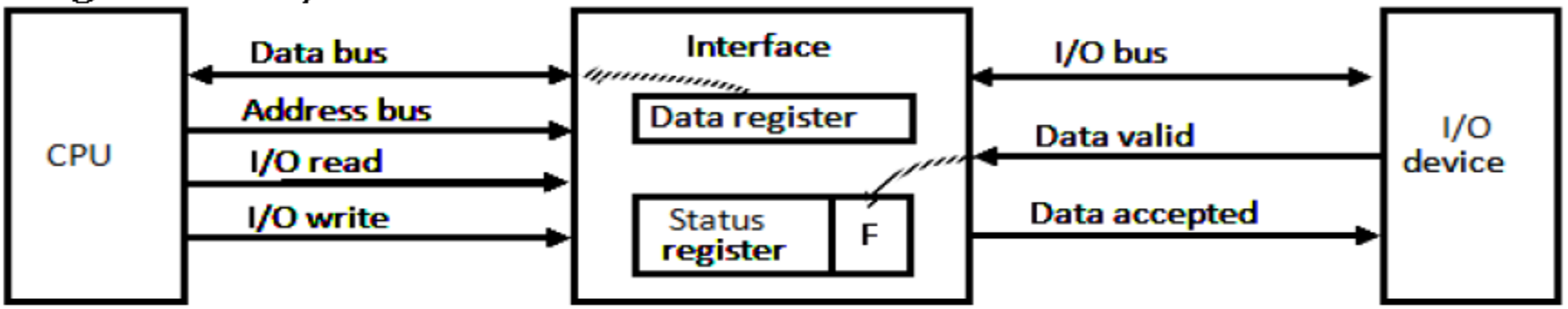


Figure 8.7: Data transfer from I/O device to CPU

- In the programmed I/O method, the I/O device does not have direct access to memory.
- An example of data transfer from an I/O device through an interface into the CPU is shown in figure 8.7.
- 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.
- The device can now disables the data valid line, but it will not transfer another byte until the data accepted line is disables by the interface.
- A program is written for the computer to check the flag in the status register to determine if a byte has been placed in the data register by the I/O device.
- This is done by reading the status register into a CPU register and checking the value of the flag bit.
- Once the flag is cleared, the interface disables the data accepted line and the device can then transfer the next data byte.

Example of Programmed I/O:

- A flowchart of the program that must be written for the CPU is shown in figure 8.8.
- 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 :

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

- Each byte is read into a CPU register and then transferred to memory with a store instruction.
- A common I/O programming task is to transfer a block of words from an I/O device and store them in a memory buffer.

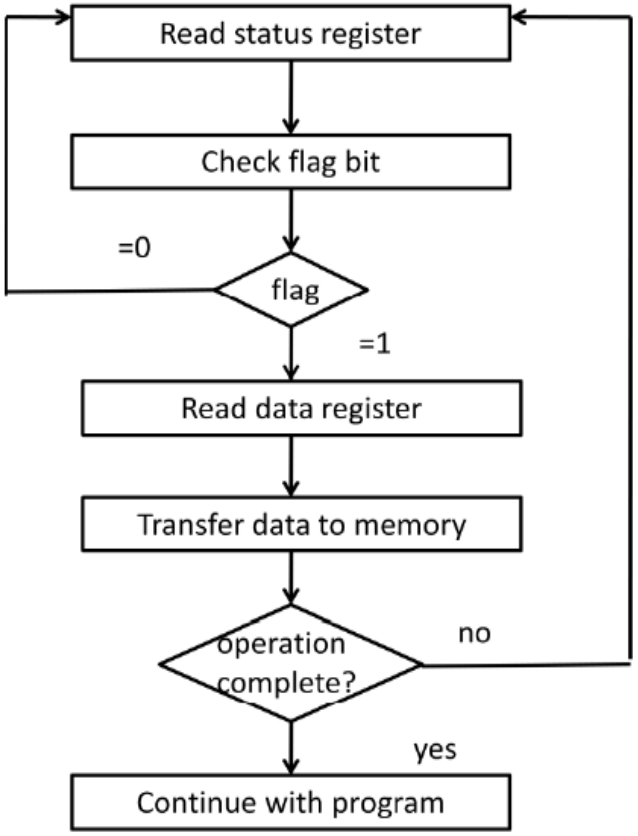


Figure 8.8: Flowchart for CPU program to input data

Write a note on Interrupt Initiated I/O

- In programmed initiated, 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.
- It can be avoided by using an interrupt facility and a special command to inform the interface to issue an interrupt request signal when data are available from the device.
- In the meantime CPU can proceed to execute another program.
- The interface meanwhile keeps monitoring the device.
- When the interface determines that the device is ready for data transfer, it generates an interrupt request to the computer.
- 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
- The way that the processor chooses the branch address of the service routine varies from one unit to another.
- In **non-vectored interrupt**, branch address is assigned to a fixed location in memory.
- In a **vectored interrupt**, the source that interrupts supplies the branch information to the computer. The information is called vector interrupt.
- In some computers the interrupt vector is the first address of the I/O service routine.
- In other computers the interrupt vector is an address that points to a location in memory where the beginning address of the I/O service routine is stored.

Priority Interrupt

What is priority interrupt? Explain Daisy Chaining.

- Determines which interrupt is to be served first when two or more requests are made simultaneously
- Also determines which interrupts are permitted to interrupt the computer while another is being serviced
- Higher priority interrupts can make requests while servicing a lower priority interrupt.

Daisy Chaining Priority

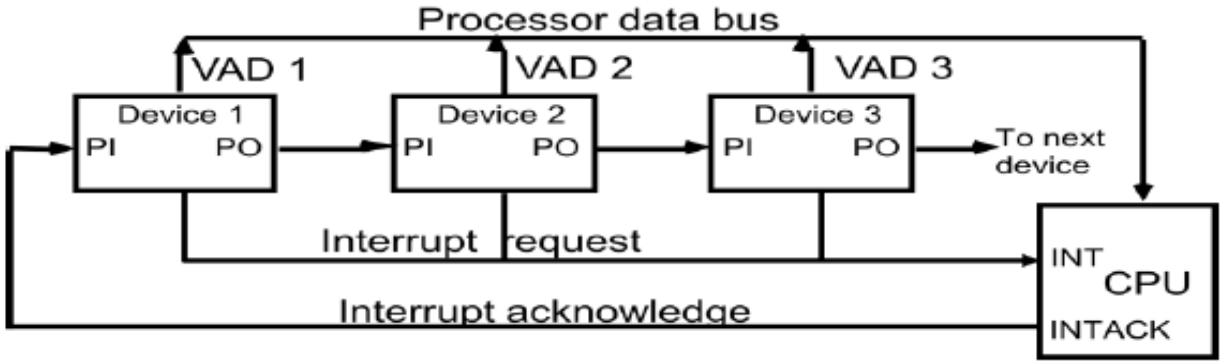


Figure 8.9: Daisy-chain priority interrupt

- The daisy-chaining method of establishing priority 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.

- This method of connection between three devices and the CPU is shown in figure 8.9.
- 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 are 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 passes on to the next device through the PO (priority out) output only if device 1 is not requesting an interrupt.
- 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 by placing a 1 in its PO output.
- 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 VAD on the data bus.
- 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.

Direct Memory Access

Write a detailed note on Direct Memory Access (DMA).

Direct Memory Access

- Transfer of data under programmed I/O is between CPU and peripheral.
- In direct memory access (DMA), Interface transfers data into and out of memory through the memory bus.
- The CPU initiates the transfer by supplying the interface with the starting address and the number of words needed to be transferred and then proceeds to execute other tasks.
- When the transfer is made, the DMA requests memory cycles through the memory bus.
- When the request is granted by the memory controller, DMA transfers the data directly into memory.

DMA controller

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

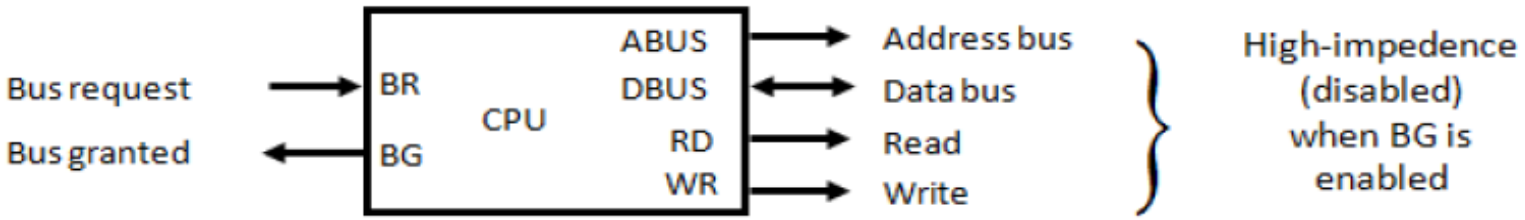


Figure 8.10: CPU bus signals for DMA transfer

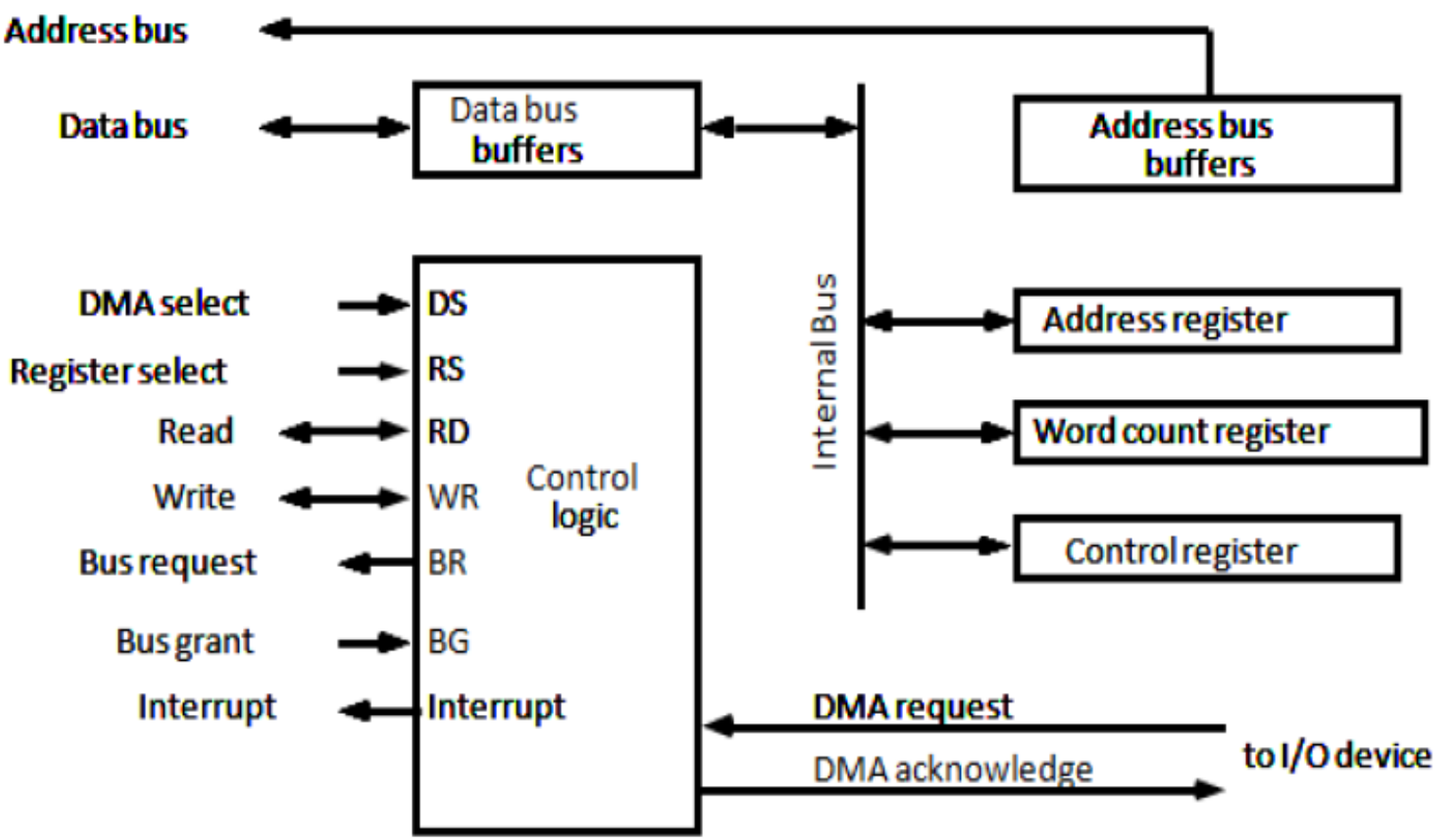


Figure 8.11: Block diagram of DMA controller

- The DMA controller needs the usual circuits of an interface to communicate with the CPU and I/O device.
- In addition, it needs an address register, a word count register, and a set of address lines.
- The address register and address lines are used for direct communication with the memory.
- The word count register specifies the number of words that must be transferred.
- The data transfer may be done directly between the device and memory under control of the DMA.
- Figure 8.11 shows the block diagram of a typical DMA controller.
- The unit communicates with the CPU via the data bus and control lines.
- The register 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 is 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 but and activating the RD or WR control.
- The DMA communicates with the external peripheral through the request and acknowledge lines by using a prescribed handshaking procedure.
- The DMA controller has three registers: an address register, a word count register, and a control register.
- 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.

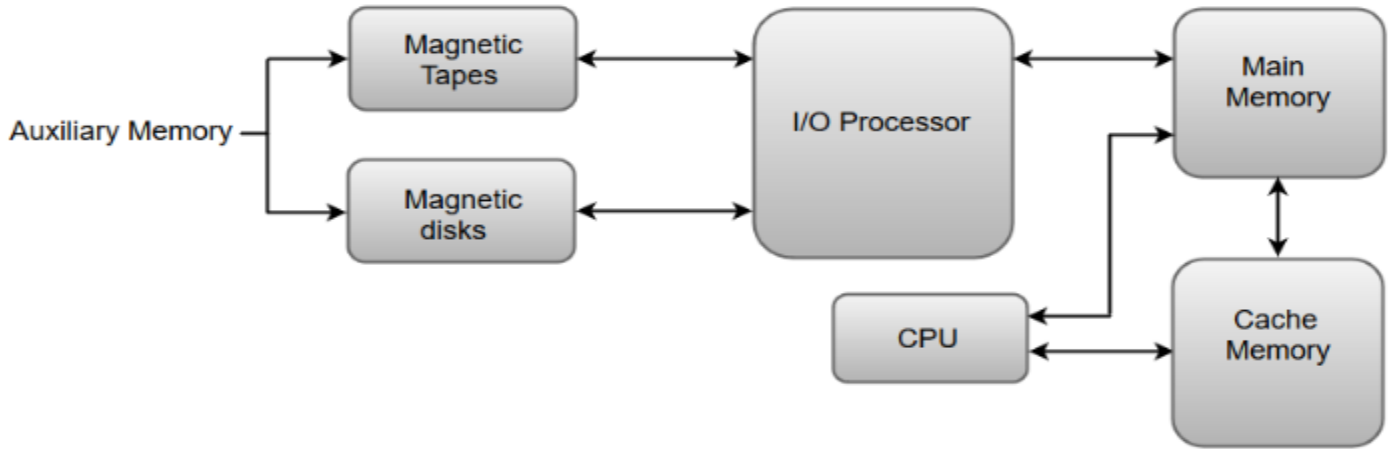
- This register is decremented by one after each word transfer and internally tested for zero.
- The control register specifies the mode of transfer.
- All registers in the DMA appear to the CPU as I/O interface registers.
- Thus the CPU can read from or write into the DMA register under program control via the data bus.
- The DMA is first initialized by the CPU.
- After that, the DMA starts and continues to transfer data between memory and peripheral unit until an entire block is transferred.
- The CPU initializes the DMA by sending the following information through the data bus
  1. The staring address of the memory block where data are available (for read) or where data are to be stored (for write)
  2. The word count, which is the number of words in the memory block.
  3. Control to specify the mode of transfer such as read or write.
  4. The starting address is stored in the address register.

Memory Organization

Explain memory hierarchy in detail.

- A memory unit is an essential component in any digital computer since it is needed for storing programs and data.
- Typically, a memory unit can be classified into two categories:
- The memory unit that establishes direct communication with the CPU is called **Main Memory**. The main memory is often referred to as RAM (Random Access Memory).
- The memory units that provide backup storage are called **Auxiliary Memory**. For instance, magnetic disks and magnetic tapes are the most commonly used auxiliary memories.

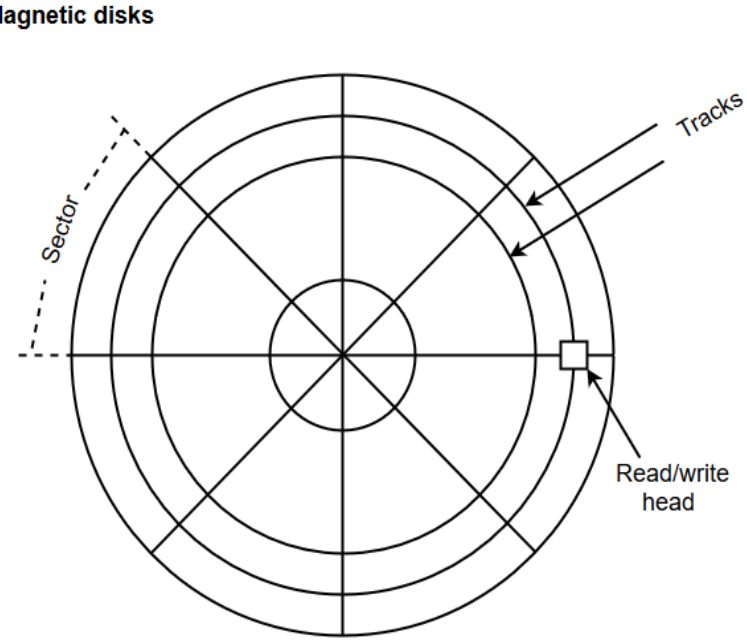
Memory Hierarchy in a Computer System:



Auxiliary memory:

- An Auxiliary memory is known as the lowest-cost, highest-capacity and slowest-access storage in a computer system.
- It is where programs and data are kept for long-term storage or when not in immediate use.
- The most common examples of auxiliary memories are magnetic tapes and magnetic disks.
- A **magnetic disk** is a digital computer memory that uses a magnetization process to write, rewrite and access data. For example, hard drives, zip disks, and floppy disks.
- A magnetic disk is a type of memory constructed using a circular plate of metal or plastic coated with magnetized materials.
- Usually, both sides of the disks are used to carry out read/write operations. However, several disks may be stacked on one spindle with read/write head available on each surface.
- The following image shows the structural representation for a magnetic disk.

- The memory bits are stored in the magnetized surface in spots along the concentric circles called tracks.
- The concentric circles (tracks) are commonly divided into sections called sectors.



- **Magnetic tape** is a storage medium that allows data archiving, collection, and backup for different kinds of data. The magnetic tape is constructed using a plastic strip coated with a magnetic recording medium.
- The bits are recorded as magnetic spots on the tape along several tracks. Usually, seven or nine bits are recorded simultaneously to form a character together with a parity bit.
- Magnetic tape units can be halted, started to move forward or in reverse, or can be rewound. However, they cannot be started or stopped fast enough between individual characters. For this reason, information is recorded in blocks referred to as records.

Main Memory :

- The main memory in a computer system is often referred to as **Random Access Memory (RAM)**. This memory unit communicates directly with the CPU and with auxiliary memory devices through an I/O processor.
- The main memory have direct connection with the CPU so, it is also known as **central memory**.
- The main memory acts as the central storage unit in a computer system. It is a relatively large and fast memory which is used to store programs and data during the run time operations.
- The primary technology used for the main memory is based on semiconductor integrated circuits.
- The programs that are not currently required in the main memory are transferred into auxiliary memory to provide space for currently used programs and data by the CPU.
- The major types of main memory are used in computer system.

1. RAM (Random Access Memory)
2. ROM (Read Only Memory)

Flash memory:

- Flash memory is **an electronic non-volatile computer memory storage medium that can be electrically erased and reprogrammed**.
- The two main types of flash memory, NOR flash and NAND flash, are named for the NOR and NAND logic gates.

I/O Processor:

- The primary function of an I/O Processor is to manage the data transfers between auxiliary memories and the main memory.

Cache Memory:

- The data or contents of the main memory that are used again and again by CPU, are stored in the cache memory so that we can easily access that data in shorter time.
- Whenever the CPU needs to access memory, it first checks the cache memory.
- If the data is not found in cache memory then the CPU moves onto the main memory. It also transfers block of recent data into the cache and keeps on deleting the old data in cache to accommodate the new one.
- Thus reducing the total execution time of the program Such a fast small memory is referred to as cache memory.
- The cache is the fastest component in the memory hierarchy and approaches the speed of CPU component
- When 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
- When the CPU refers to memory and finds the word in cache, it is said to produce a **hit** Otherwise, it is a **miss**
- The performance of cache memory is frequently measured in terms of a quantity called hit ratio

  Hit ratio = hit / (hit+miss)

Virtual Memory

- Virtual memory is a feature of an operating system that enables a computer to be able to compensate (balance) shortages of physical memory by transferring pages of data from random access memory to disk storage.
- This process is done temporarily and is designed to work as a combination of RAM and space on the hard disk.
- A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory.**
- Virtual memory serves two purposes:

  1) it allows us to extend the use of physical memory by using disk.

  2) it allows us to have memory protection, because each virtual address is translated to a physical address.
- Virtual memory is commonly implemented by demand paging and segmentation system.

Explain Content Addressable Memory (CAM).

- 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.
- The block diagram of an associative memory is shown in figure 9.3.

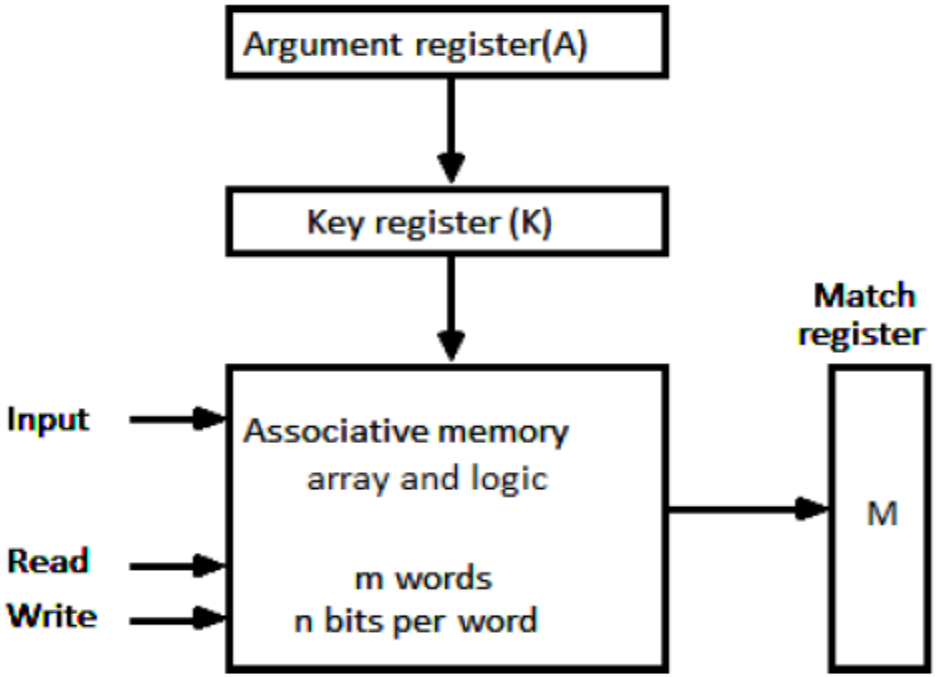


Figure 9.3: Block diagram of associative memory

- It consists of a memory array and logic form 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.
- Reading is accomplished by a sequential access to memory for those words whose corresponding bits in the match register have been set.

Hardware Organization

- 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^{st} in their corresponding position of the key register are compared.
- Thus the key provides a mask or identifying piece of information which specifies how the reference to memory is made.
- To illustrate with a numerical example, suppose that the argument register A and the key register K have the bit configuration shown below.
- Only the three leftmost bits of A are compared with memory words because K has 1's in these position.

A	101 111100	
K	111 000000	
$Word_1$	100 111100	no match
$Word_2$	101 000001	match

- Word 2 matches the unmasked argument field because the three leftmost bits of the argument and the word are equal.

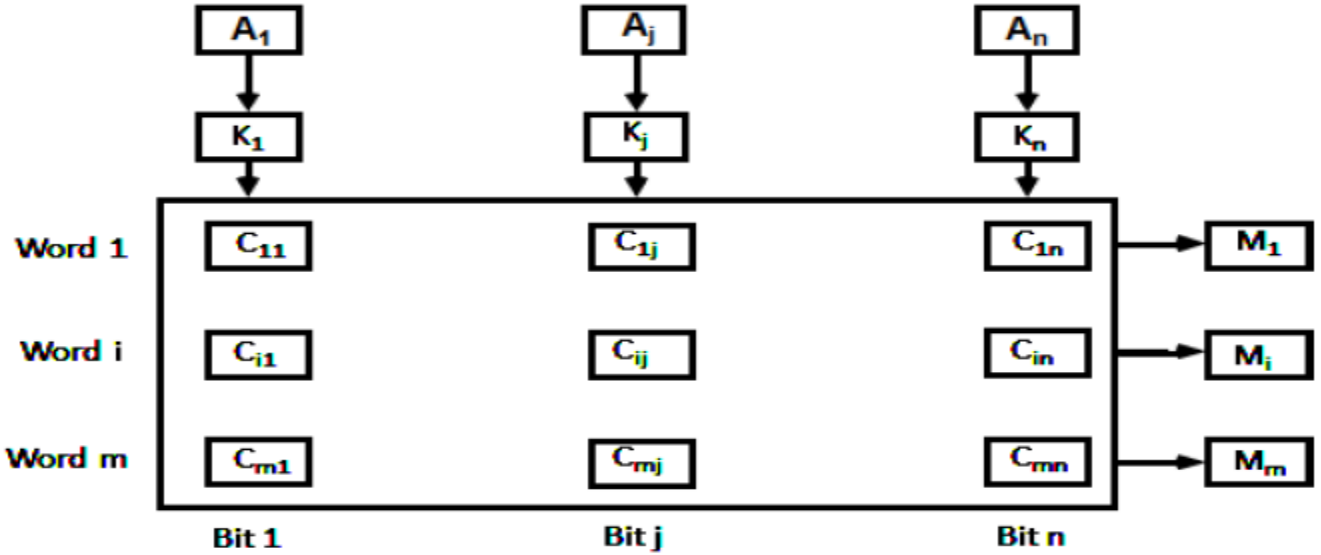


Figure 9.4: Associative memory of *m* word, n cells per word.

- The relation between the memory array and external registers in an associative memory is shown in figure 9.4.
- 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 words 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.

Define Cache memory. Discuss associative mapping in organization of cache memory.

Cache memory

- Cache is a fast small capacity memory that should hold those information which are most likely to be accessed.
- The basic operation of the cache is, 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.
- The transformation of data from main memory to cache memory is referred to as a **mapping process**.

Associative mapping

- Consider 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 miss, the CPU reads the word from main memory and the word is then transferred to cache.

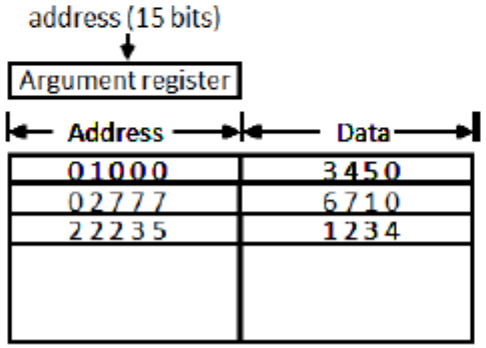


Figure 9.5: Associative mapping cache (all numbers in octal)

- 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 figure 9.5 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.
- If the address is found the corresponding 12-bit data is read and sent to CPU.
- If no match occurs, the main memory is accessed for the word.
- The address data pairs 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.
- This constitutes a first-in first-one (FIFO) replacement policy.

Discuss direct mapping in organization of cache memory.

- 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 from the tag field.
- The figure 9.6 shows that main memory needs an address that includes both the tag and the index.

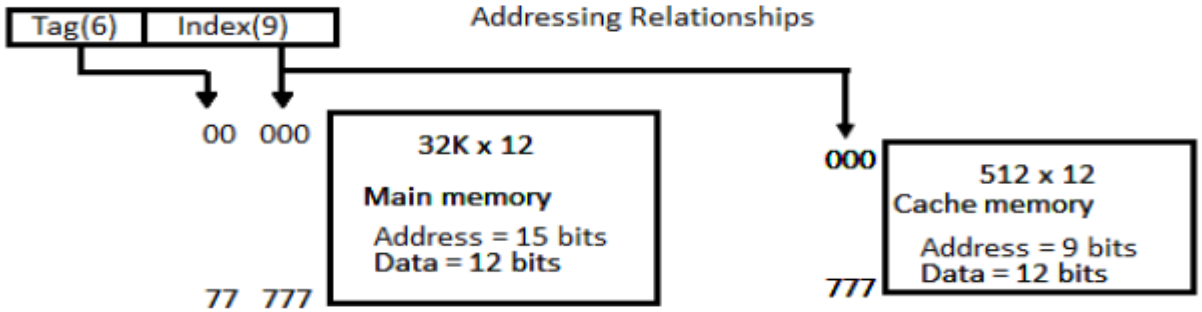


Figure 9.6: Addressing relationships between main and cache memories

- The number of bits in the index field is equal to the number of address bits required to access the cache memory.
- The internal organization of the words in the cache memory is as shown in figure 9.7.

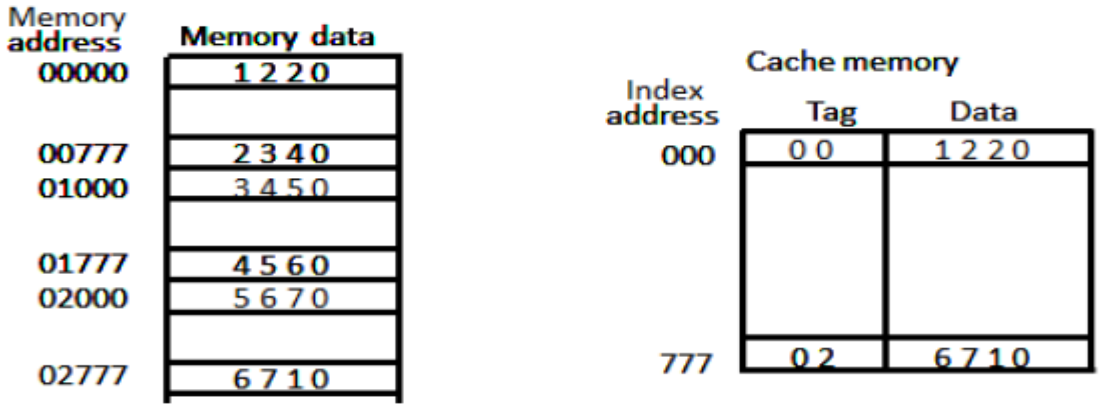


Figure 9.7: Direct mapping cache organization

- 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 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 word at address zero is presently stored in the cache (index = 000, tag = 00, data = 1220).
- Suppose that the CPU now wants to access the word at address 02000.
- The index address is 000, so it is used to access the cache. The two tags are then compared.
- The cache tag is 00 but the address tag is 02, which does not produce a match.
- Therefore, the main memory is accessed and the data word 5670 is transferred to the CPU.
- The cache word at index address 000 is then replaced with a tag of 02 and data of 5670.
- The **disadvantage** of direct mapping is that two words with the same index in their address but with different tag values cannot reside in cache memory at the same time.

Discuss set-associative mapping in organization of cache memory.

- A third type of cache organization, called set associative mapping 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 one set-associative cache organization for a set size of two is shown in figure 9.8.

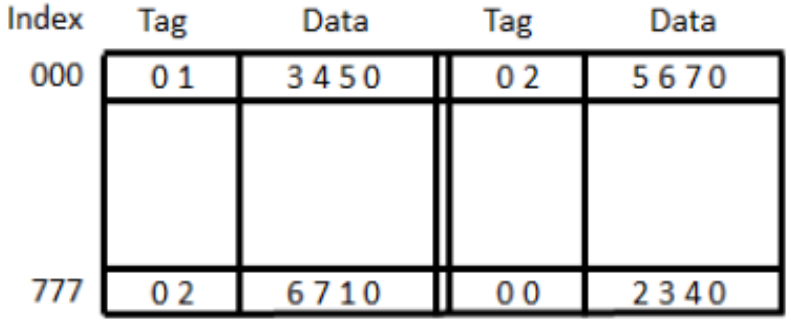


Figure 9.8: Two-way set-associative mapping cache

- Each index address refers to two data words and their associated terms.
- Each tag required 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 × 36. It can accommodate 1024 words or main memory since each word of cache contains two data words.

- In generation a set-associative cache of set size k will accommodate K word of main memory in each word of cache.
- The octal numbers listed in figure 9.8 are with reference to the main memory contents.
- 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".
- 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).