MODULE 4
INPUT/OUTPUT ORGANIZATION

There are a number of input/output (**I/O**) devices, that can be connected to a computer. The input maybe from a keyboard, a sensor, switch, mouse etc. Similarly output may be a speaker, monitor, printer, a digital display etc.

These variety of I/O devices exchange information in varied format, having different word length, transfer speed is different, but are connected to the same system and exchange information with the same computer. Computer must be capable of handling these wide variety of devices.

ACCESSING I/O-DEVICES

A **single bus-structure** can be used for connecting I/O-devices to a computer. The simple arrangement of connecting set of I/O devices to memory and processor by means of system bus is as shown in the figure. Such an arrangement is called as Single Bus Organization.

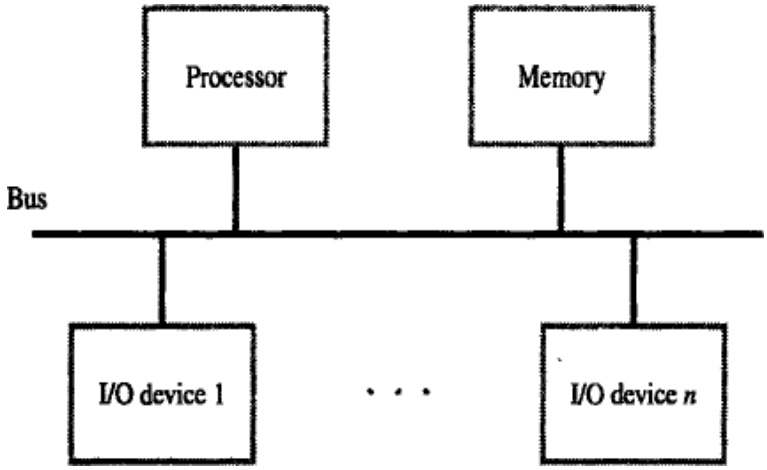


Fig: A Single Bus structure

- The single bus organization consists of
    - Memory
    - Processor
    - System bus
    - I/O device
- The **system bus** consists of **3 types of buses:**
    - Address bus (Unidirectional)
    - Data bus (Bidirectional)
    - Control bus (Bidirectional)
- The system bus enables all the devices connected to it to involve in the data transfer operation.
- The system bus establishes data communication between I/O device and processor.
- Each I/O device is assigned a unique set of address.
- When processor places an address on address-lines, the intended-device responds to thecommand.
- The processor requests either a read or write-operation.
- The requested data are transferred over the data-lines

Steps for input operation:

- The address bus of system bus holds the address of the input device.
- The control unit of CPU generates $\overline{\text{IORD}}$ Control signal.
- When this control signal is activated the processor reads the data from the input device(DATAIN) into the CPU register.

Steps for output operation:

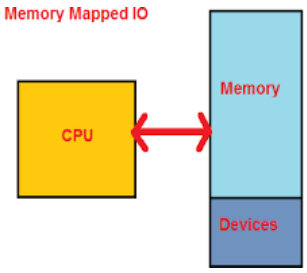
- The address bus of system bus holds the address of the output device.
- The control unit of CPU generates $\overline{\text{IOWR}}$ control signal.
- When this control signal is enabled CPU transfers the data from processor register to outputdevice(DATAOUT)

There are 2 schemes available to connect I/O devices to CPU

1. **Memory mapped I/O:**

- In this technique, both memory and I/O devices use **the common bus** to transfer the data to CPU .
- same address space is used for both memory and I/O interface. They have only one set of read and write signals.
- All memory related instructions are used for data transfer between I/O and processor.
- In case of memory mapped I/O input operation can be implemented as,MOVE DATAIN , R0

  DATAIN → Source, R0 → destination



This instruction sends the contents of location DATAIN to register R0.

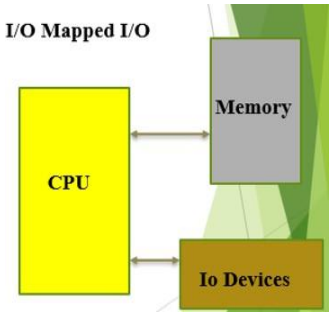
- Similarly output can be implemented as, MOVE R DATAOUT

  R → Source, DATAOUT → Destination

- The data is written from R0 to DATAOUT location (address of output buffer.

2) **I/O Mapped I/O:**

- In this technique, a separate bus is used for I/O devices and memory to transfer the data to CPU. Address space for memory and I/O devices are different.
- Hence two sets of instruction are used for data transfer.
- One set for memory operations and another set for I/O operations.Whole address space is available for the program.
- Eg – IN AL, DX



Memory Mapped I/O	I/O Mapped I/O
Memory & I/O share the entire address range of processor	Processor provides separate address range for memory & I/O
Processor provides more address lines for accessing memory	Less address lines for accessing I/O
More Decoding is required	Less decoding is required
Memory control signals used to control Read & Write I/O operations	I/O control signals are used to control Read & Write I/O operations

I/O INTERFACE

The hardware arrangement of connecting i/p device to the system bus is as shown in the fig.

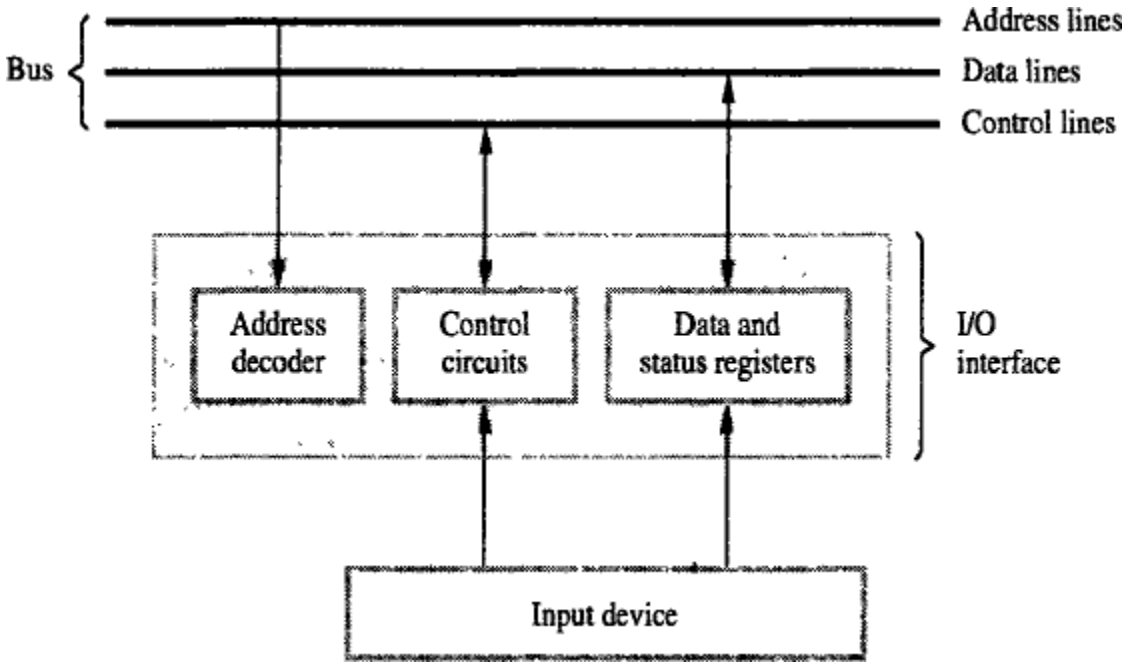


Fig: I/O interface for an input device

This hardware arrangement is called as I/O interface. The I/O interface consists of 3 functional devicesnamely:

1) **Address Decoder:**

- Its function is to decode the address, in-order to recognize the input device whose address isavailable on the unidirectional address bus.
- The recognition of input device is done first, and then the control and data registers becomes active.
- The unidirectional address bus of system bus is connected to input of the address decoder asshown in figure

2) **Control Circuit:**

- The control bus of system bus is connected to control circuit as shown in the fig.
- The processor sends commands to the I/O system through the control bus.
- It **controls the read write operations** with respect to **I/O device**.

3) **Status & Data register:**

- It specifies type of operation (either read or write operation) to be performed on I/O device. Itspecifies the position of operation.

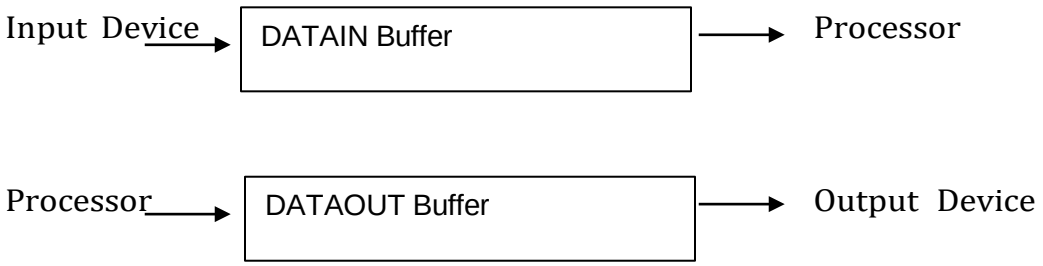
4) **Data Register:**

- The data bus carries the data from the I/O devices to or from the processor. The data bus isconnected to the data/ status register.
- The data register stores the data, read from input device or the data, to be written into outputdevice. There are 2 types:

DATAIN - Input-buffer associated with keyboard.

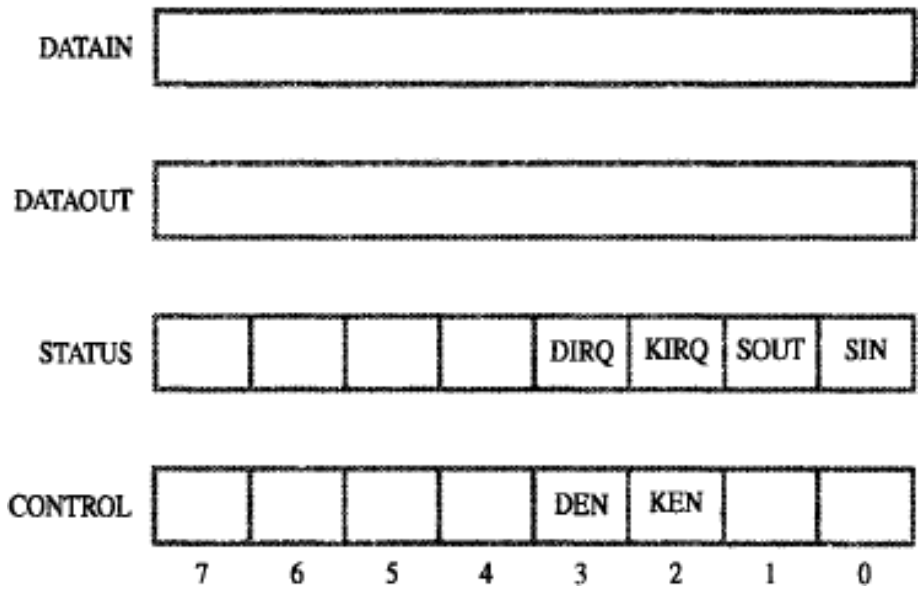
DATAOUT -Output data buffer of a display/printer.

Data buffering is an essential task of an I/O interface. Data transfer rates of processor and memory are high, when compared with the I/O devices, hence the data are buffered at the I/O interface circuit and then forwarded to output device, or forwarded to processor in case of input devices.



Input & Output registers –

Various registers in keyboard and display devices -



DATAIN register: is a part of input device. It is used to store the ASCII characters read from keyboard.

DATAOUT register: is a part of output device. It is used to store the ASCII characters to bedisplayed on the output device.

STATUS register stores the status of working of I/O devices –

- **SIN flag – This flag is set to 1**, when **DATAIN buffer contains the data** from keyboard. **The flag is set to 0**, after the data is passed from DATAIN buffer to the processor.
- **SOUT flag – This flag is set to 1**, when **DATAOUT buffer is empty** and the data can be added to it by processor. The flag is set to 0, when DATAOUT buffer has the data to be displayed.
- **KIRQ (Keyboard Interrupt Request)** – By setting this flag to 1, keyboard requests the processor to obtain its service and an interrupt is sent to the processor. It is used along with the SIN flag.

- **DIRQ(Display Interrupt Request)** – The output device request the processor to obtain its service for output operation, by activating this flag to 1.

Control registers

KEN (keyboard Enable) – Enables the keyboard for input operations.

DEN (Display Enable) – Enables the output device for input operations.

Program Controlled I/O

- In this technique **CPU is responsible for executing data from the memory for output and storing data in memory for executing of Programmed I/O**
- **Drawback** of the Programmed I/O: was that the **CPU has to monitor the units all the times when the program is executing.** 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 keeping an eye to the executing of program.

- It is the process of controlling the input and output operations by executing 2 sets of instruction,one set for input operation and the next set for output operation.

- The program checks the status of I/O register and reads or displays data. Here the I/O operationis controlled by program.

```
WAITK      TestBit #0, STATUS      (Checks SIN
           flag)Branch = 0 WAITK
           Move DATAIN, R0         (Read character)[
           *Code to read a character from DATAIN to R0]
```

This code checks the SIN flag, and if it is set to 0 (ie. If no character in DATAIN Buffer), then move back to WAITK label. This loop continues until SIN flag is set to 1. When SIN is 1, data ismoved from DATAIN to R0 register. Thus the program, continuously checks for input operation.

Similarly code for Output operation,

```
WAITD      TestBit #0, STATUS      (Checks SOUT flag)
           Branch = 0 WAITD
           Move R0, DATAOUT        (Send character for
                                   display)
```

The code checks the SOUT flag, and if it is set to 1 (ie. If no character in DATAOUT Buffer), then move back to WAITK label. This loop continues until SOUT flag is set to 0. When SOUT is 0, data is moved from R0 register to DATAOUT (ie. Sent by processor).

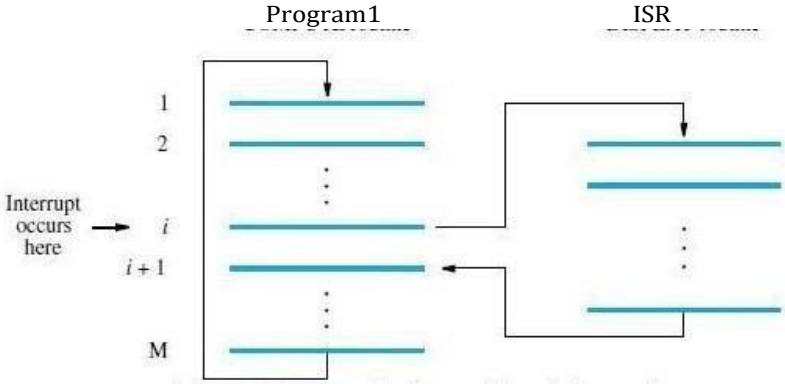
Illustrate a program that reads one line from the keyboard, stores it in memory buffer, and echoes it back to the display in an I/O interfaces.

```
        Move        #LINE,R0          Initialize memory pointer.
WAITK   TestBit     #0,STATUS         Test SIN.
        Branch=0    WAITK             Wait for character to be entered.
        Move        DATAIN,R1         Read character.
WAITD   TestBit     #1,STATUS         Test SOUT.
        Branch=0    WAITD             Wait for display to become ready.
        Move        R1,DATAOUT        Send character to display.
        Move        R1,(R0)+          Store charater and advance pointer.
        Compare     #$0D,R1           Check if Carriage Return.
        Branch≠0    WAITK             If not, get another character.
        Move        #$0A,DATAOUT      Otherwise, send Line Feed.
        Call        PROCESS           Call a subroutine to process the
                                        the input line.
```

Figure 4.4 A program that reads one line from the keyboard, stores it in memory buffer, and echoes it back to the display.

Interrupt

- It is an event which suspends the execution of one program and begins the execution of another program.
- In program controlled I/O, a program should continuously check whether the I/O device is free. By this continuous checking the processor execution time is wasted. It can be avoided by I/O device **sending an 'interrupt' to the processor, when I/O device is free**.
- The interrupt invokes a subroutine called **Interrupt Service Routine (ISR)**, which resolves the cause of interrupt.
- The occurrence of interrupt causes the processor to transfer the execution control from user program to ISR.



The following steps takes place when the interrupt related instruction is executed:

- It suspends the execution of current instruction i.
- Transfer the execution control to sub program from main program.
- Increments the content of PC by 4 memory location.
- It decrements SP by 4 memory locations.
- Pushes the contents of PC into the stack segment memory whose address is stored in SP.
- It loads PC with the address of the first instruction of the sub program.

The following steps takes place when '*return*' instruction is executed in ISR -

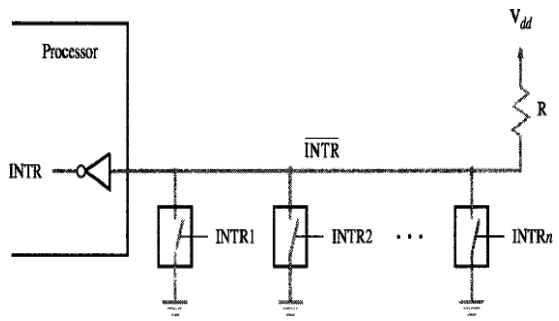
- It transfers the execution control from ISR to user program.
- It retrieves the content of stack memory location whose address is stored in SP into the PC.
- After retrieving the return address from stack memory location into the PC it increments theContent of SP by 4 memory location.

Interrupt Latency / interrupt response time is the delay between the time taken for receiving aninterrupt request and start of the execution of the ISR.

Generally, the long interrupt latency is unacceptable.

INTERRUPT HARDWARE

- The external device (I/O device) sends interrupt request to the processor by activating a bus lineand called as interrupt request line.
- All I/O device uses the same single interrupt-request line.
- One end of this interrupt request line is connected to input power supply by means of a register.
- The another end of interrupt request line is connected to INTR (Interrupt request) signal ofprocessor as shown in the fig.



- The I/O device is connected to interrupt request line by means of switch, which is grounded asshown in the fig.
- When all the switches are open the voltage drop on interrupt request line is equal to the V_{DD} andINTR value at process is 0.
- This state is called as in-active state of the interrupt request line.
- The I/O device interrupts the processor by closing its switch.

- When switch is closed the voltage drop on the interrupt request line is found to be zero, as theswitch is grounded, hence INTR=0 and $\overline{INTR}$=1.
- The signal on the interrupt request line is logical OR of requests from the several I/O devices.Therefore, INTR=$\overline{INTR1}$ + $\overline{INTR2}$ + + $\overline{INTRn}$

ENABLING AND DISABLING THE INTERRUPTS

The arrival of interrupt request from external devices or from within a process, causes the suspension ofon-going execution and start the execution of another program.

- Interrupt arrives at any time and it alters the sequence of execution. Hence the interrupt to beexecuted must be selected carefully.
- All computers can enable and disable interruptions as desired.
- When an interrupt is under execution, other interrupts should not be invoked. This is performedin a system in different ways.
- The problem of infinite loop occurs due to successive interruptions of active INTR signals.

- There are **3 mechanisms to solve problem of infinite loop**:
  1) Processor should ignore the interrupts until execution of first instruction of the ISR.
  2) Processor should automatically disable interrupts before starting the execution of the ISR.
  3) Processor has a special INTR line for which the interrupt-handling circuit.

     Interrupt-circuit responds only to leading edge of signal. Such line is called edge-triggered.
- Sequence of events involved in **handling an interrupt-request:**
  1) The device raises an interrupt-request.
  2) The processor interrupts the program currently being executed.
  3) Interrupts are disabled by changing the control bits in the processor status register (PS).
  4) The device is informed that its request has been recognized. And in response, the device deactivates the interrupt-request signal.
  5) The action requested by the interrupt is performed by the interrupt-service routine.
  6) Interrupts are enabled and execution of the interrupted program is resumed.

HANDLING MULTIPLE DEVICES

While handling multiple devices, the issues concerned are:

- How can the processor recognize the device requesting an interrupt?
- How can the processor obtain the starting address of the appropriate ISR?
- Should a device be allowed to interrupt the processor while another interrupt isbeing serviced?
- How should 2 or more simultaneous interrupt-requests be handled?

VECTORED INTERRUPT

- A device requesting an interrupt identifies itself by sending a special-code to processor over bus.
- Then, the processor starts executing the ISR.
- The special-code indicates starting-address of ISR.
- The special-code length ranges from 4 to 8 bits.
- The location pointed to by the interrupting-device is used to store the staring address to ISR.

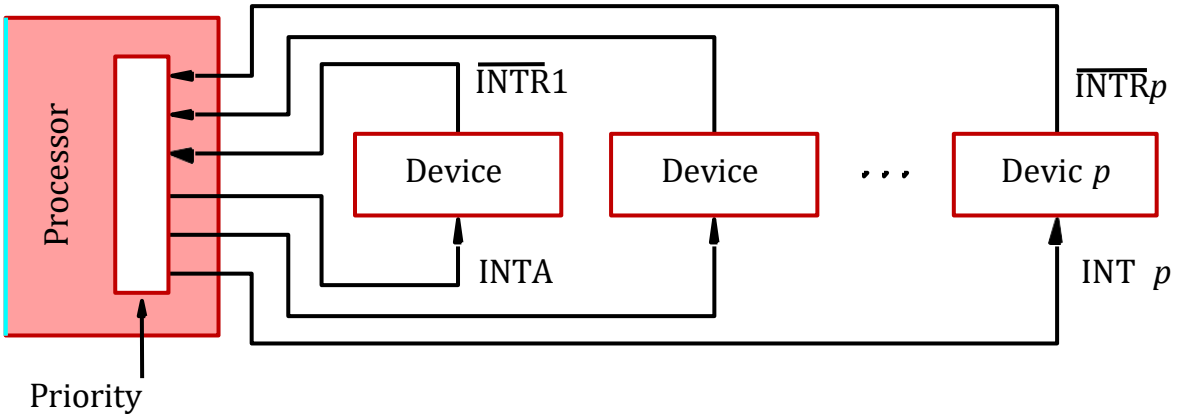
- The staring address to ISR is called the **interrupt vector**.
- Processor

  → loads interrupt-vector into PC &

  → executes appropriate ISR.
- When processor is ready to receive interrupt-vector code, it activates INTA line.
- Then, I/O-device responds by sending its interrupt-vector code & turning off the INTR signal.
- The interrupt vector also includes a new value for the Processor Status Register

INTERRUPT NESTING

- A multiple-priority scheme is implemented by using separate INTR & INTA lines for each device
- Each INTR line is assigned a different priority-level as shown in Figure.



- Each device has a separate interrupt-request and interrupt-acknowledge line.
- Each interrupt-request line is assigned a different priority level.
- Interrupt requests received over these lines are sent to a priority arbitration circuit in the processor.
- If the interrupt request has a higher priority level than the priority of the processor, then the request is accepted.
- Priority-level of processor is the priority of program that is currently being executed.
- Processor accepts interrupts only from devices that have higher-priority than its own.

- At the time of execution of ISR for some device, priority of processor is raised to that of the device.
- Thus, interrupts from devices at the same level of priority or lower are disabled.

Privileged Instruction

- Processor's priority is encoded in a few bits of PS word. (PS = Processor-Status).
- Encoded-bits can be changed by **Privileged Instructions** that write into PS.
- Privileged-instructions can be executed only while processor is running in **Supervisor Mode**.
- Processor is in supervisor-mode only when executing operating-system routines.

Privileged Exception

- User program cannot

  → accidently or intentionally change the priority of the processor &

  → disrupt the system-operation.

- An attempt to execute a privileged-instruction while in user-mode leads to a **Privileged Exception**.

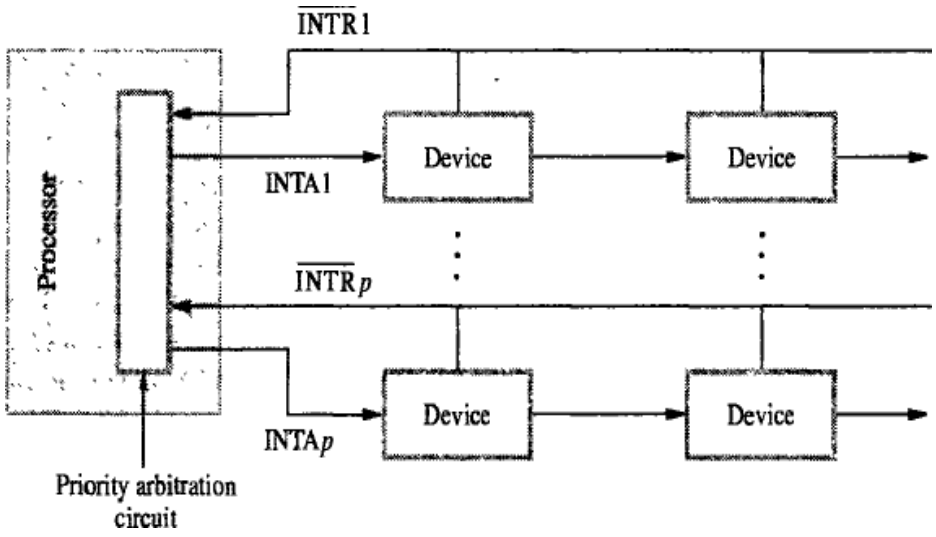
SIMULTANEOUS REQUESTS

DAISY CHAIN

- The daisy chain with multiple priority levels is as shown in the figure.
- The interrupt request line INTR is common to all devices as shown in the fig.
- The interrupt acknowledge line is connected in a daisy fashion as shown in the figure.
- This signal propagates serially from one device to another device.
- The several devices raise an interrupt by activating INTR signal. In response to the signal, processortransfers its device by activating INTA signal.
- This signal is received by device 1. The device-1 blocks the propagation of INTA signal to device-2,when it needs processor service.
- The device-1 transfers the INTA signal to next device when it does not require the processor service.
- In daisy chain **arrangement device-1 has the highest priority.**

Advantage: It requires fewer wires than the individual connection

ARRANGEMENT OF PRIORITY GROUPS



- In this technique, devices are organizes in a group and each group is connected to the processor at adifferent priority level.
- With in a group devices are connected in a daisy chain fashion as shown in the figure.

Direct Memory Access (DMA)

- It is the process of **transferring the block of data at high speed in between main memory and external device (I/O devices) without continuous intervention of CPU is called as DMA.**
- The DMA operation is performed by one control circuit and is part of the I/O interface.
- This control circuit is called DMA controller. Hence DMA transfer operation is performed by DMAcontroller.
- To initiate Directed data transfer between main memory and external devices DMA controller needsparameters from the CPU.
- These 3 Parameters are:

1)Starting address of the memory block. 2)No of words to be transferred.
3)Type of operation (Read or Write).

After receiving these 3 parameters from CPU, DMA controller establishes directed data transferoperation between main memory and external devices without the involvement of CPU.

- **Register of DMA Controller:**

It consists of 3 type of register:

Starting address register:

The format of starting address register is as shown in the fig. It is used to store the starting addressof the memory block.

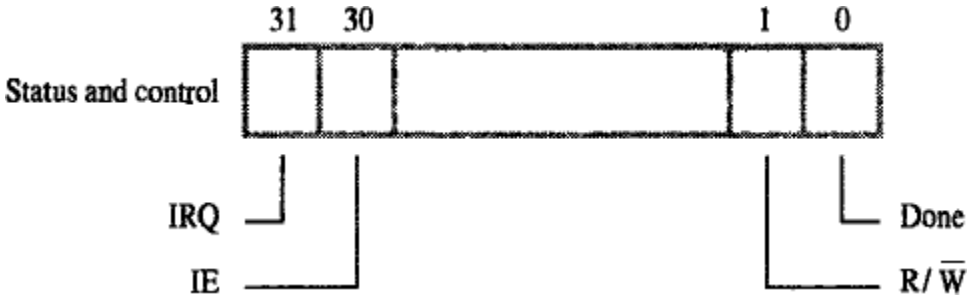
.

Word-Count register:

The format of word count register is as shown in fig. It is used to store the no of words to be transferredfrom main memory to external devices and vice versa.

Status and Controller register:

The format of status and controller register is as shown in fig.



a) **DONE bit:**

- The DMA controller sets this bit to 1 when it completes the direct data transfer between mainmemory and external devices.
- This information is informed to CPU by means of DONE bit.

b) **R/W (Read or Write):**

- This bit is used to differentiate between memory read or memory write operation.
- The R/W = 1 for read operation and
  = 0 for write operation.
- When this bit is set to 1, DMA controller transfers the one block of data from external deviceto main memory.
- When this bit is set to 0, DMA controller transfers the one block of data from main memoryto external device.

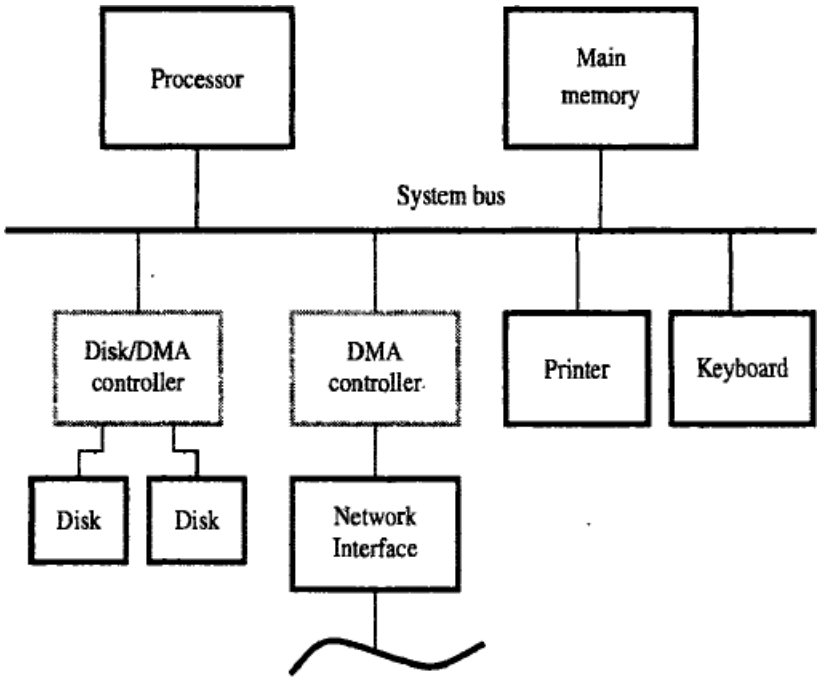
c) IE (Interrupt enable) bit:

- The DMA controller enables the interrupt enable bit after the completion of

DMA operationd)**Interrupt request (IRQ):**

- The DMA controller requests the CPU to transfer new block of data from source todestination by activating this bit.

The computer with DMA controller is as shown in the fig.:



- The DMA controller connects two external devices namely disk 1 and disk 2 to system bus asshown in the above fig.
- The DMA controller also interconnects high speed network devices to system bus as shownin the above fig.
- Let us consider direct data transfer operation by means of DMA controller without the involvement of CPU in between main memory and disk 1 as indicated by dotted lines (in the fig.).

- To establish direct data transfer operation between main memory and disk
  1. DMA controller request the processor to obtain **3 parameters** namely:
     1) **Starting address of the memory block.**
     2) **No of words to be transferred.**
     3) **Type of operation (Read or Write).**

- After receiving these 3 parameters from processor, DMA controller directly transfers block ofdata main memory and external devices (disk 1).

- This information is informed to CPU by setting respective bits in the status and controllerregister of DMA controller.
  These are 2 types of request with respect to
  system bus1). CPU request.
  2). DMA request.
  Highest priority will be given to DMA request.

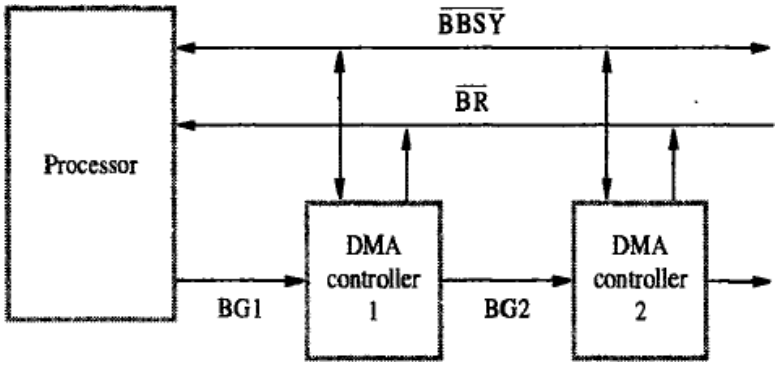
- Actually the CPU generates memory cycles to perform read and write operations. The DMA controller steals memory cycles from the CPU to perform read and write operations. This approach is called as **"Cycle stealing".**

- An exclusive option will be given for DMA controller to transfer block of data from external devices to main memory and from main memory to external devices. This technique is called as "**Burst mode of operation.**"

BUS ARBITRATION

- Any device which initiates data transfer operation on bus at any instant of time is called as Bus-Master.
- When the bus mastership is transferred from one device to another device, the next device isready to obtain the bus mastership.
- The bus-mastership is transferred from one device to another device based on the principle ofpriority system. There are two types of bus-arbitration technique:

a) Centralized bus arbitration:
In this technique CPU acts as a bus-master or any control unit connected to bus can be acts as a busmaster.



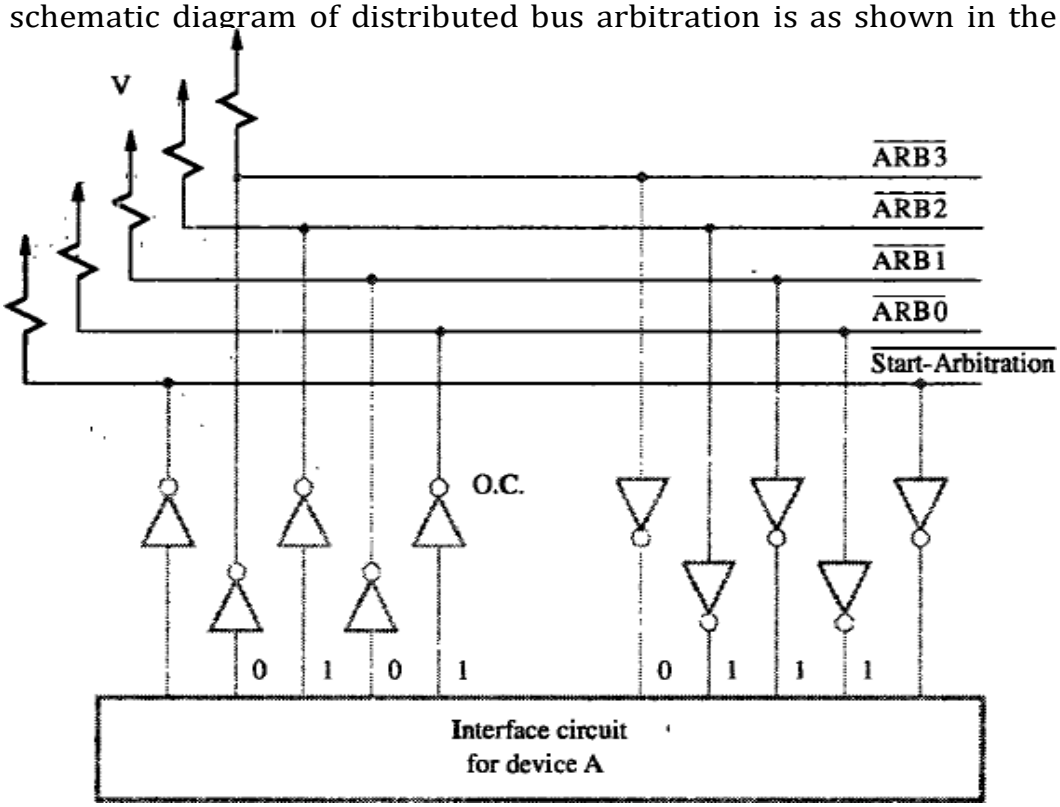
The schematic diagram of **centralized bus arbitration** is as shown in the fig.:

The following steps are necessary to transfer the bus mastership from CPU to one of the DMAcontroller:

- The DMA controller request the processor to obtain the bus mastership by activating BR (Busrequest) signal
- In response to this signal the CPU transfers the bus mastership to requested devices DMAcontroller1 in the form of BG (Bus grant).
- When the bus mastership is obtained from CPU the DMA controller1 blocks the propagation of busgrant signal from one device to another device.
- The BG signal is connected to DMA controller2 from DMA controller1 in as daisy fashion style isas shown in the figure.
- When the DMA controller1 has not sent BR request, it transfers the bus mastership to DMAcontroller2 by unblocking bus grant signal.
- When the DMA controller1 receives the bus grant signal, it blocks the signal from passing to DMA controller2 and enables BBSY signal. When BBSY signal is set to 1 the set of devices connected to system bus doesn't have any rights to obtain the bus mastership from the CPU.

b) Distributed bus arbitration:

- In this technique 2 or more devices trying to access system bus at the same time may participatein bus arbitration process.
- The schematic diagram of distributed bus arbitration is as shown in the figure



- The external device requests the processor to obtain bus mastership by enabling start arbitrationsignal.
- In this technique 4 bit code is assigned to each device to request the CPU in order to obtain busmastership.
- Two or more devices request the bus by placing 4 bit code over the system bus.
- The signals on the bus interpret the 4 bit code and produces winner as a result from the CPU.
- When the input to the one driver = 1, and input to the another driver = 0, on the same bus line,this state is called as "Low level voltage state of bus".
- Consider 2 devices namely A & B trying to access bus mastership at the same time.Let assigned code for devices A & B are 5 (0101) & 6 (0110) respectively.
- The device A sends the pattern (0101) and device B sends its pattern (0110) to master. The signals on the system bus interpret the 4 bit code for devices A & B produces device B as a winner.
- The device B can obtain the bus mastership to initiate direct data transfer between external devices and main memory.

The Memory System

Speed, Size and Cost

The block diagram of memory hierarchy is as shown in the figure below.

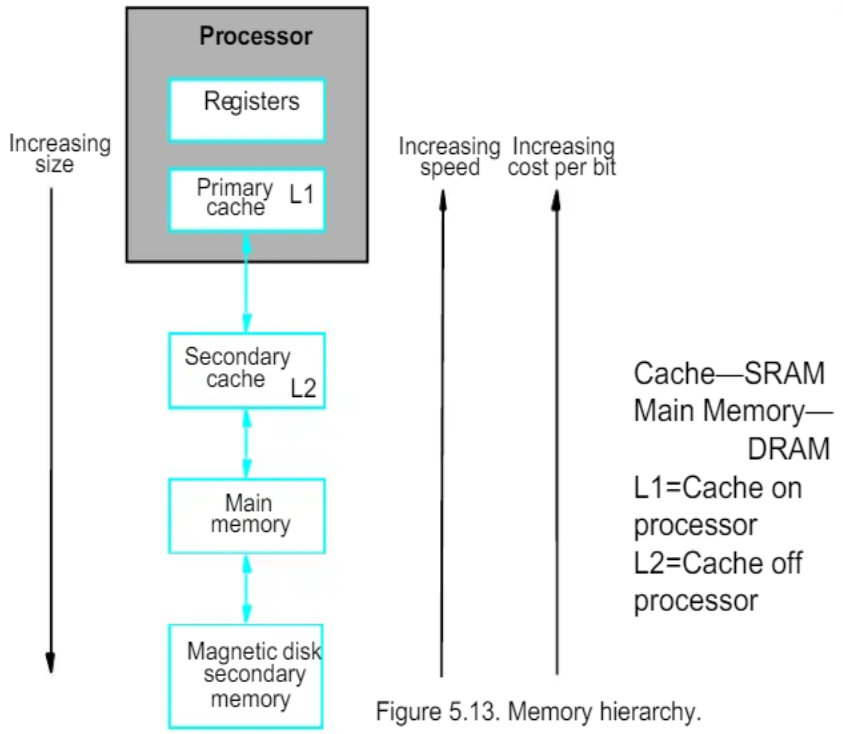


Figure 5.13. Memory hierarchy.

- Registers: The fastest access is to data held in registers. Hence registers are part of the memory hierarchy. More speed, small size and cost per bit is also more.
- At the next level of hierarchy, small amount of memory can be directly implemented on the processor chip.
- This memory is called as **processor cache**. **It holds the copy of recently accessed data and instructions.**

There are **2 levels of caches viz level-1 and level-2.**

Level-1 cache is **part of the processor** and **level-2 cache** is placed in **between level-1 cache and main memory.**

- The level-2 cache is implemented using SRAM chips
- The next level in the memory hierarchy is called as **main memory**. It is implemented using dynamic memory components (DRAM). **The main memory is larger but slower than cache memory.** The access time for main memory is ten times longer than the cache memory
- The level next in the memory hierarchy is called as **secondary memory**. It **holds huge amount of data.**

Characteristics	SRAM	DRAM	Magnetis Disk
Speed	Very Fast	Slower	Much slower than DRAM
Size	Large	Small	Small
Cost	Expensive	Less Expensive	Low price

Memory	Speed	Size
Registers	Very high	Lower
Primary cache	High	Lower
Secondary cache	Low	Low
Main memory	Lower than Seconadry cache	High
Secondary Memory	Very low	Very High

- The **main-memory is built with DRAM**
- **SRAMs are used in cache memory, where speed is essential.**
- The Cache-memory is of 2 types:

  1) Primary/Processor Cache (Level1 or L1 cache)

  ➢ It is always located on the processor-chip.

  2) Secondary Cache (Level2 or L2 cache)

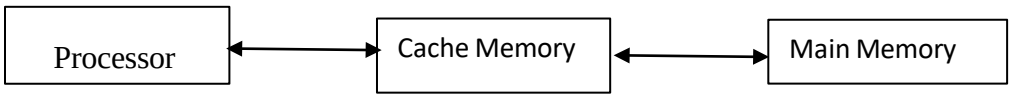
  ➢ It is placed between the primary-cache and the rest of the memory.

- The memory is implemented using the dynamic components (SIMM, DIMM).

The access time for main-memory is about 10 times longer than the access time for L1cache.

Cache Memory

It is the **fast access memory located in between processor and main memory**



as shown inthe fig. It is designed to reduce the access time.

The cache memory holds the copy of recently accessed data and instructions.

- The processor needs less access time to read the data and instructions from the cache memory as compared to main memory .
- Hence by incorporating cache memory, in between processor and main memory, itis possible to enhance the performance of the system.
- The effectiveness of cache mechanism is based on the property of "**Locality of Reference".**

Locality of Reference

- Many instructions in the localized areas of program are executed repeatedly during sometime of execution
- Remainder of the program is accessed relatively infrequently
- There are 2 types of locality reference:

1) **Temporal**

- The recently executed instructions are likely to be executed again and again.
- Eg –instruction in loops, nested loops and few function calls.

2) **Spatial**

- Instructions in close proximity to recently executed instruction are likely to be executed soon. (near by instructions)

- If active segment of program is placed in cache-memory, then total execution time can be reduced.
- **Cache Block / cache line** refers to the set of contiguous address locations of some size.
- The Cache-memory stores a reasonable number of blocks at a given time.

- This number of blocks is small compared to the total number of blocks available in main-memory.
- Correspondence b/w main-memory-block & cache-memory-block is specified by **mapping-function.**
- If the cache memory is full, one of the block should be removed to create space forthe new block, this is decided by **cache control hardware.**
- The collection of rule for selecting the block to be removed is called the **Replacement Algorithm.**
- The cache control-circuit determines whether the requested-word currently exists in the cache.
- If data is available, for read-operation, the data is read from cache.
- The **write-operation** (writing to memory) is done in **2 ways:**
  1) **Write-through protocol &**
  2) **Write-back protocol.**

Write-Through Protocol

- Here the cache-location and the main-memory-locations are updated simultaneously.

Write-Back Protocol

- This technique is to

  → update only the cache-location &

  → mark the cache-location with a flag bit called **Dirty/Modified Bit.**

- The word in memory will be updated later, when the marked-block is removed from cache.

During Read-operation

- If the requested-word currently does not exists in the cache, then **read-miss** will occur.
- To overcome the read miss, *Load–through/Early restart protocol* is used.

Load–Through Protocol

- The block of words that contains the requested-word is copied from the memory into cache.
- After entire block is loaded into cache, the requested-word is forwarded to processor.

➢

During Write-operation

- If the requested-word does not exists in the cache, then **write-miss** will occur.
  1) If **Write Through Protocol** is used, the information is written directlyinto main-memory.
  2) If **Write Back Protocol** is used,

     → then block containing the addressed word is first brought into the cache&

     → then the desired word in the cache is over-written with the newinformation.

Mapping functions

There are 3 techniques to map main memory blocks into cache memory –

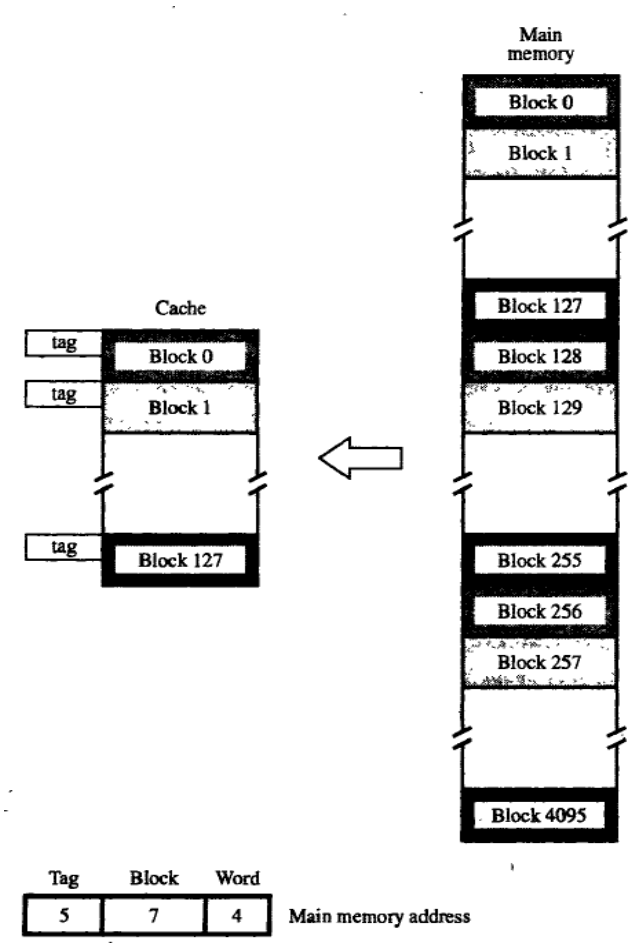
1. Direct mapped cache
2. Associative Mapping
3. Set-Associative Mapping

DIRECT MAPPING

- The simplest way to determine cache locations in which to store memory blocks is the direct mapping technique as shown in the figure.

- Cache block number= (block-j of main memory)%128;

- If there are 128 blocks in a cache, the block-j of the main-memory maps onto block-jmodulo-128 of the cache . When the memory-blocks 0, 128, & 256 are loaded into cache, the block is stored in cache-block 0. Similarly, memory-blocks 1, 129, 257 are stored in cache-block 1.(eg:1mod 128=1, 129 mod 128=1)
- The contention may arise
  1) Even when the cache is full.
  2) But more than one memory-block is mapped onto a given cache-block position.
- The contention is resolved by allowing the new blocks to overwrite the currently resident-block. Memory-address determines placement of block in the cache.



main memory block has to be placed in particular cache block number by using below formula
Cache block number=main memory block number % number of blocks present in cache memory.

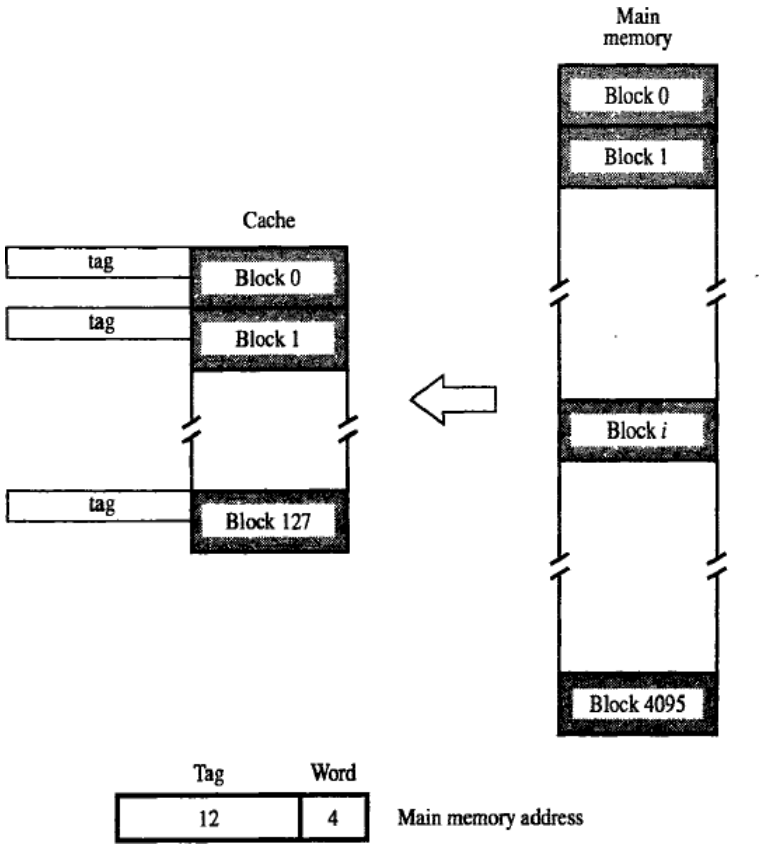
For eg: main memory block 129 has to be placed in cache block number 1 by using above formula i.e
Cache block number=129 % 128 (consider remainder that is 1).
Cache block number=258 % 128 (consider remainder that is 2).
Main memory block 258 has to be placed in cache block 2

- The main memory block is loaded into cache block by means of **memory address.** The main memory address consists of 3 fields as shown in the figure.
- **Each block consists of 16 words**. Hence least significant **4 bits** are **used to select one of the 16 words.**
- The **7bits of memory address are used to specify the position of the cache block**, location. The most significant **5 bits of the memory address are stored in the tag bits**. **The tag bits are used to map one of 2^5 = 32 blocks into cache block location** (tag bit has value 0-31).
- **The higher order 5 bits of memory address are compared with the tag bits** associated with cache location**. If they match**, then the desired word is in that block of the cache.
- **If there is no match**, then the block containing the required word must first be read from the main memory

and loaded into the cache. It is very easy to implement, but not flexible.

2. Associative Mapping:

- It is also called as associative mapped cache. It is much more flexible.
- In this technique main memory block can be placed into **any cache block position**.
- In this case , 12 tag bits are required to identify a memory block when it is resident of the cache memory.
- The Associative Mapping technique is illustrated as shown in the fig.



- In this technique 12 bits of address generated by the processor are compared with the tag bits of each block of the cache to see if the desired block is present. This is called as associative mapping technique.

3.Set Associative Mapping:

- It is the combination of **direct and associative mapping techniques.**
- The blocks of cache are divided into several groups. Such a groups are called as **sets.**
- Each set consists of number of cache blocks. A memory block is loaded into one of the cache sets.
- The **main memory address consists of three fields**, as shown in the figure.
- The **lower 4 bits of memory address** are used to **select a word from a 16 words.**
- A cache consists of **64 sets** as shown in the figure. Hence **6 bit set field** is used to select a **cache set from 64 sets.**
- As there are 64 sets, the memory is divided into groups containing 64 blocks, where each group is given a tag number.
- The most significant **6 bits of memory address is compared with the tag fields of each set to determine whether memory block is available or not.**
- The following figure clearly describes **the working principle of Set Associative Mapping technique.**

- cache that has "k" blocks per set is called as "**k-way set associative cache**".
- Each block contains a control-bit called a **valid-bit**.
- The Valid-bit indicates that whether the block contains valid-data (updated data).
- The dirty bit indicates that whether the block has been modified during its cache residency.

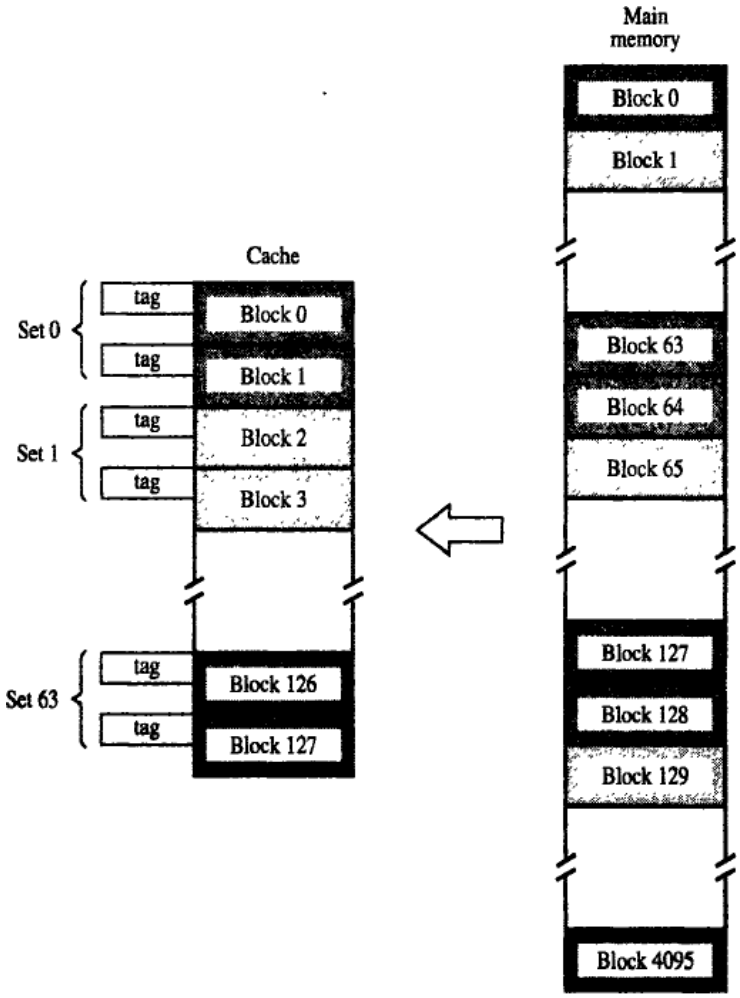
Valid-bit=0 - When power is initially applied to system.

Valid-bit=1 - When the block is loaded from main-memory at first time.

- If the main-memory-block is updated by a source & if the block in the source is already exists in the cache, then the valid-bit will be cleared to "0".
- If Processor & DMA uses the same copies of data then it is called as **Cache Coherence Problem**.
- **Advantages:**

1) Contention problem of direct mapping is solved by having few choices for block placement.

2) The hardware cost is decreased by reducing the size of associative search.



Tag	Set	Word
6	6	4

Main memory address