9 Memory Organization

The memory unit is an essential component of any digital computer system since it is needed for storing programs and data. Memory unit is capable of storing the programs and the data that are needed to perform a particular task. A CPU should have rapid, uninterrupted access to these memories so that it can operate at or near its maximum speed. The goal of every memory system is to provide adequate storage capacity with acceptable level of performance and cost. There is not enough space in one memory unit to accomodate all the programs used in a typical computer, not all the accumulated information is required by the processor at the smae time. Therefore, it is more economical to use low-cost storage devices to serve as a backup for storing the information that is not currently used by the CPU. Following all the some criteria on which we decide which memory unit is to be used.

1. Cost
2. Speed
3. Memory access time
4. Data transfer rate
5. Reliability
6. Memory cycle time.

In this chapter, we will discuss about various memory device organisation. There are various types of memories some provides good speed but they are costly, some provide less speed but they are cheaper so we discuss the organisation of various memory units.

9.1. MEMORY HIERARCHY

Memory in a computer system is required for storage and subsequent retrieval of the instructions and data. A computer system uses variety of devices for storing these instructions and data which are required for its operations. Normally we classify the information to be stored on computer in two basic categories: Data and the Instructions.

"The storage devices along with the algorithm or information on how to control and manage thes storage devices constitute the memory system of a computer."

A memory system is a very simple system yet it exhibits a wide range of technology and types The basic objective of a computer system is to increase the speed of computation. Likewise the basi objective of a memory system is to provide fast, uninterrupted access by the processor to the memory such that the processor can operate at the speed it is expected to work.

But does this kind of technology where there is no speed gap between processor and memory speed exist? The answer is yes they do, but unfortunately as the access time (time taken by CPU to access a location in memory) becomes less and less the cost per bit of memory becomes increasingly higher. In addition, normally these memories require power supply till the information need to be stored. Both these things are not very convenient, but on the other hand the memories with smaller cost have very high access time which will result in slower operation of the CPU. Thus, the cost vs access time anomaly has lead to a hierarchy of memory where we supplement fast memories with larger, cheaper, slower memories. These memory units may have very different physical and operational characteristics, therefore, making the memory system very diverse in type, cost, organisation, technology and performance. This memory hierarchy will work only if the frequency of access to the slower memories are significantly less than the faster memories.

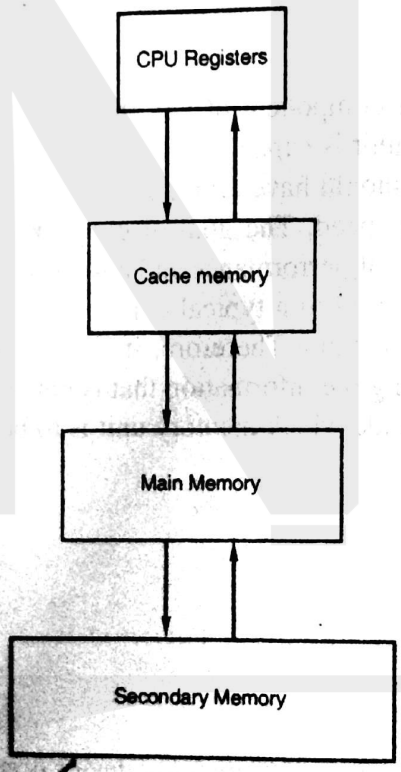

Fig. 9.1. The Memory Hierarchy

Thus, a memory system can be considered to consist of three group of memories. These are:

(a) **Internal Processor Memories:** These consist of the small set of high speed registers which are internal to a processor and are used as temporary locations where actual processing is done.

(b) **Primary Memory or Main Memory:** It is a large memory which is fast but not as fast as internal processor memory. This memory is accessed directly by the processor. It is mainly based on integrated circuits.

(c) **Secondary Memory/Auxiliary Memory/Backing Store:** Auxiliary memory in fact is much larger in size than main memory but is slower than main memory. It normally stores system programs

(programs which are used by system to perform various operational functions), other instructions, programs and data files. Secondary memory can also he used as an overflow memory in case the main memory capacity has been exceeded. Secondary memories cannot be accessed directly by a processor. First the information of these memories is transferred to the main memory and then the information can be accessed as the information of main memory.

There is another kind of memory which is increasingly being used in modern computers, this is called Cache memory. It is logically positioned between the internal memory (registers) and main memory. It stores or catches some of the content of the main memory which is currently in use of the processor. We will discuss about this memory in greater details in a subsequent section of this unit.

Before discussing more about these memories let us first discuss the technological terms commonly used in defining and accessing the memory.

9.2 CHARACTERISTICS/PARAMETERS

9.3. MAIN MEMORY

The main memory is the control storage device of a computer system. It is relatively large and fast memory that is used to store the data and programs that are currently needed by CPU. The principal technology behind the main memory is based on semiconductor integrated circuits. Integrated circuit RAM chips are available in two possible operating modes, static and dynamics. The main difference between SRAM and DRAMs is how their bit cells all constructed. The dynamic RAM provides reduced power consumption and larger storage capacity in a single dynamic RAM provides reduced power consumption and larger storage capacity in a single memory chip. The static RAM is easier to use and has shorter read and write cycles. Main memory is made up of RAM and ROM chips. RAM is used to store the programs and data that is to be changed while ROm is used for storing the programs that are permanently resident in the computer.

ROM portion of the main memory is needed for storing the initial program that is known as bootstrap program that runs on the computer when power is turned on. The contents of ROM remains constant when power is turned on and off. RAM and ROM chips are available in a variety of sizes. When memory requirement is large then we combine a number of chips to form the required memory size.

9.3.1. RAM and ROM Chips

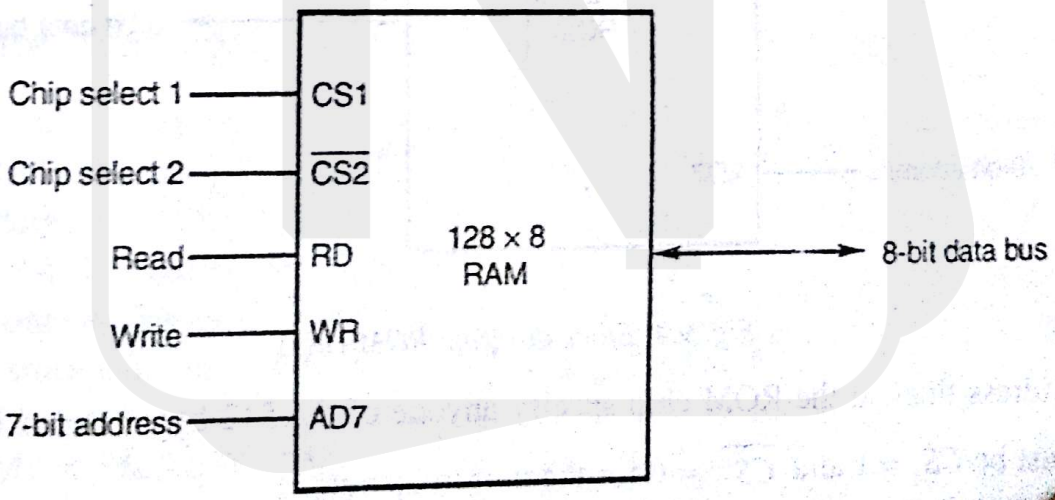

Fig. 9.3. RAM chip

Figure 9.3 depicts a typical RAM chip. RAM chips are better suited for communicating with the CPU. It has bi-directional data bus that allows the transfer of data in both the direction. A bidirectional bus can be constructed using three state buffers. A three state buffer output can be placed in one of the possible three states. The capacity of RAM chip is 128 × 8 means there are 128 words each word is of 8 bits.

For 128 words we require an address line of 7 bits since $2^7 = 128$. The read and write inputs specify the memory operation and the two Chips Selects (CS) control inputs are for enabling the chp only when it is selected by the microprocessor. The function table is listed in Table 9.1.

Table 9.1. Function Table

CS_1	$\overline{CS_2}$	RD	WR	Memory function	State of data bus
0	0	×	×	Inhibit	High impedance
0	1	×	×	Inhibit	High impedance
1	0	0	0	Inhibit	High Impedance
1	0	0	1	Write	Input data to RAM
1	0	1	×	Read	Output data from RAM
1	1	×	×	Inhibit	High impedance

The unit is in operations only when $CS_1 = 1$ and $\overline{CS_2} = 0$. In this combination the memory can be placed in a write or read mode. When WR input is enabled, a byte is stored from the data bus into a location that is specified by the address input lines. When RD input is enabled, the content of the selected byte is placed into the data bus.

A ROM chip is organised in a similar manner. Read Only Memories (ROMs) are memory devices that the CPU can read but cannot write. Computers use them for holding permanent information, for holding the constant values that specify the system configuration. Since a ROM can only read, the data bus can only be in an output mode. The block diagram is shown in Fig. 9.4.

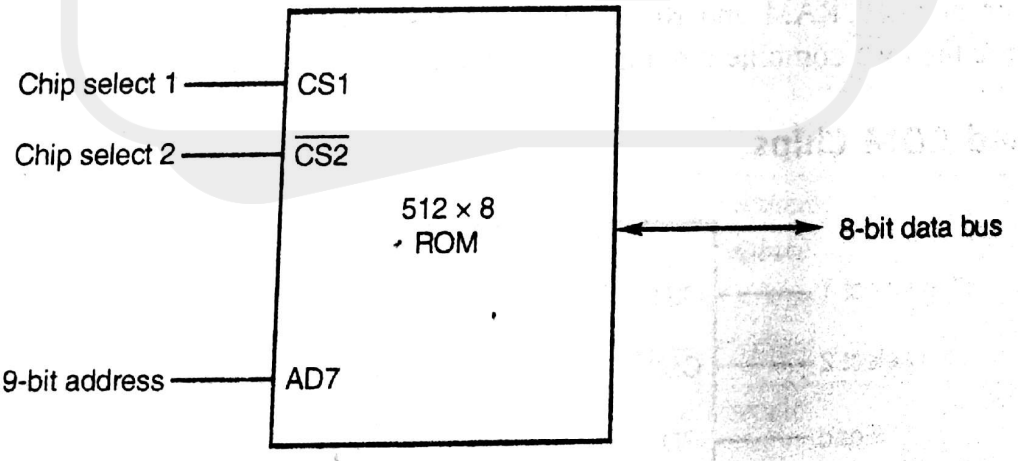

Fig. 9.4. Block diagram ROM chips

The nine address lines in the ROM chip specify anyone of the 512 bytes stored in it. The two chip select inputs must be $CS_1 = 1$ and $\overline{CS_2} = 0$ for the unit to operate.

9.4.2. Cache Memory

The active portions of the program and data are placed in fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program. Such a fast small memory is referred to as a cache memory.

9.4.2.1. Cache Memory System

A cache memory system includes a small amount of fast memory (SRAM) and a large amount of slow memory (DRAM). This system is configured to simulate a large amount of fast memory. Figure 9.15 shows a cache memory system.

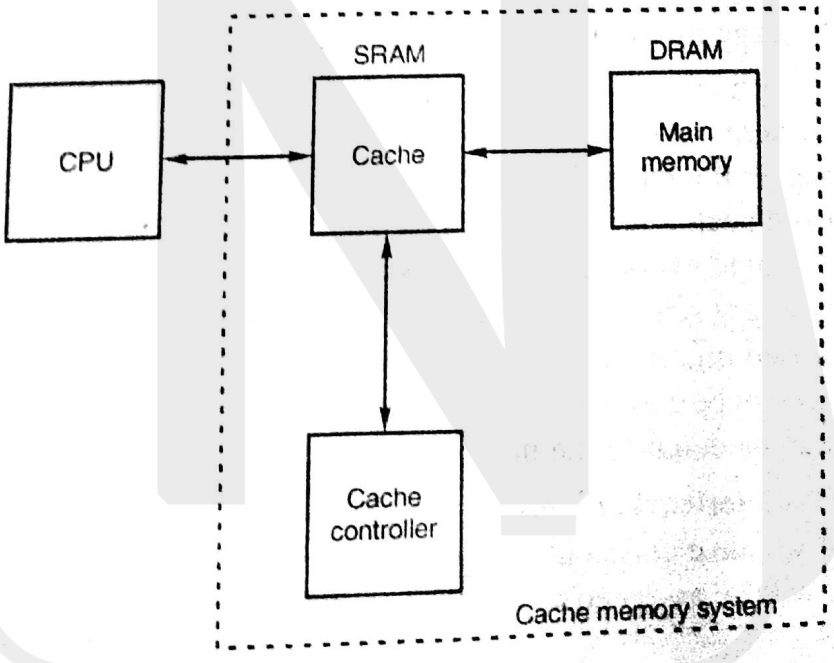

Fig. 9.15. Cache memory system.

The access time of cache memory is very less as compared to the access time of main memory. When a read request is received from the CPU, the contents of a block of memory words containing the location specified are transferred into the cache one word at a time. Subsequently, when the program references any of the location in this block, the desired contents are read directly from the cache. Usually, the cache memory can store a reasonable number of blocks at any given time, but this number of small compared to the total number of blocks in the main memory. If the processor finds that the dressed data is not available in cache, then processor accesses that data from the main memory (DRAM). The percentage of accesses where the processor finds the data word it needs in the cache memory is called the hit rate. The hit rate is normally greater than 90 percent.

$$\text{Hit rate} = \frac{\text{Number of hits}}{\text{Number of read / write bus cycle}} \times 100\%$$

Program Locality

In cache memory system, prediction of memory location for the next access is essential. This is possible because computer systems usually access memory from the consecutive location. This prediction of next memory address from the current memory address is known as program locality. Program locality enables cache contrller.

Locality of Reference

We know that a program may contain a simple loop, nested lops or a few procedures that repeatedly call each other. The actual detailed pattern of instruction sequencing is not important, the point is that many instructions in localised areas of the program are executed repeatedly during some time period, and the remainder of the program is accessed relatively infrequently. this is referred to as locality of reference. It manifests itself in two ways : *temporal* and *spatial*. The temporal means that a recently executed instruction is likely to be executed again very soon. The spatial means that instructions stored near by to the recently executed instructions are also likely to be executed soon.

The temporal aspect of the locality reference suggests that whenever an information of instruction and data is first needed, this information should be brought into cache where it will hope fully remain until it is needed again. The spatial aspects suggests that instead of bringing just one item from the main memory to the cache, it is wise to bring several items that reside at adjacent addresses as well. We use the term block to refer to a set of continuous addresses of size.

9.4.2.2. Element of Cache Design

The cache design elements include cache size, mapping function, replacement algorithm, write policy, block size and number of caches.

Cache Size

The size of the cache should be small enough so that the overall average cost per bit is close to that of main memory alone and large enough so that the overall average access time is close to that of the cache alone.

Mapping Function

The cache memory can store a reasonable number of blocks at any given time, but this number is small compared to the total number of blocks in the main memory. Thus, we have to use mapping

functions to relate the main memory blocks and cache blocks. There are two mapping functions commonly used : Direct mapping and associative mapping.

Replacement Algorithm

When the cache is full and a memory word that is not in the cache is referenced. The cache controller must decide which block should be removed to create space for the new block that contains the referenced word. The collection of rules for making this decision constituents the replacement algorithm. There are four most common replacement algorithms :

- Least-Recently Used (LRU)
- First-In First-Out (FIFO)
- Least-Frequently-Used (LFU)
- Random

Write Policy

It is also known as cache updating policy. In cache system, two copies of the same data can exist at a item, one in cache and one in main memory. If one copy is altered and other is not two different sets of data become associated with the same address. To prevent this the cache system has updating system such as : Write through system, buffered write through system, and write-back system.

Block Size

It should be optimum for cache memory system.

Number of Caches

When on-cip cache is insufficient the secondary cache is used. The cache design changes as number of caches used in the system changes.

9.4.2.3. Mapping Function

The transformation of data from main memory to cache memory is referred to as a mapping process. The basic characteristics 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 cache memory is referred to as a mapping process. There are two mapping techniques which decides the cache organisation :

1. Direct mapping technique
2. Associative mapping technique

The associative mapping technique is faster classified as fully fassociative and set associative technique.

To discuss the mapping technique of cache mapping, we consider the following specifications.

Memory	*Memory capacity*	*Number of block per block*	*Number of words*
1. Cache	2048 (2K)	128	16
2. Main memory	64 K	4 K	16

The group of 128 blocks in main memory form a page. Next we discuss the mapping technique in the following sections.

(a) Direct Mapping

The main memory blocks can be placed in one and only one placed in the cache, this mapping is called as direct mapping. The direct mapping technique is the simplest mapping technique. In this technique, each block from the main memory has only one possible location (fixed place) in the cache organisation. The block *j* of the main memory maps onto block j modules 128 of the cache, illustrated in the Fig. 9.16. Therefore, whenever one of the main memory blocks 0, 128, 256 ... (the first block in each page of main memory is stored in the first block of cache memory), it is stored in cache block 0. Blocks 1, 129, main memory is stored in the first block of cache memory), it is stored in cache block 0. Blocks 1, 129, main memory is stored in the first block of cache memory), it is stored in cache block 0. Blocks 1, 129, 257, ... are stored in cache block 1, and so on.

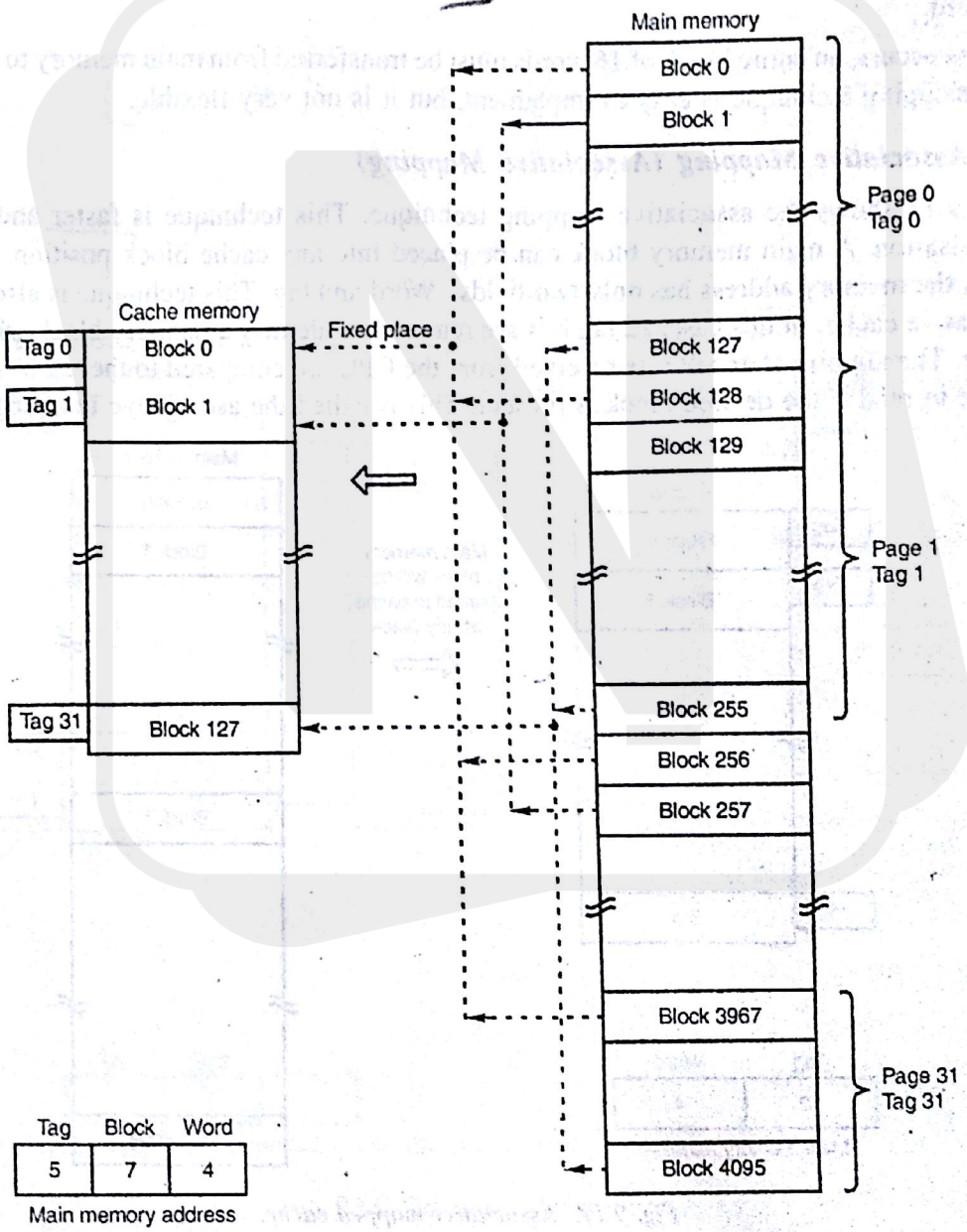

Fig. 9.16. Direct mapped cache.

The placement of a block in the cache is determined from the address of main memory. The address is divided into three fields as shown in Fig. 9.16. The lower order 4-bits select one of the 16 words in a block. This field is known as word field. The second field known as block field is used to distinguish a block from other blocks. When a new block enters the cache, the 7 bit cache block field determines the position in which this block must be stored. This third field is a tag field. Each word in cache consists of the data word and its associated tag field.

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 word field and block fields are 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 (read from cache memory) and the desired data word is in cache. If there is no match, there is miss (data is not available in the cache, data read from main memory) and the required word read from main memory. It is then stored in the cache together with need tag by replacing the previous word.

The miss occurs, an entire block of 16 words must be transferred from main memory to cache memory. The direct mapping technique is easy to implement, but it is not very flexible.

(b) Fully Associative Mapping (Associative Mapping)

Figure 9.17 shows the associative mapping technique. This technique is faster and must flexible cache organisation. A main memory block can be placed into any cache block position. As there is no fixed block, the memory address has only two fields : Word and tag. This technique is also referred to as fully associative cache. In this case, 12 tag bits are reuqired to identify a memory block when it is placed in the cache. The tag bits of an address received from the CPU are compared to the tag bits of each block of the cache to read if the desired block is present. This is called the asosciative mapping technique.

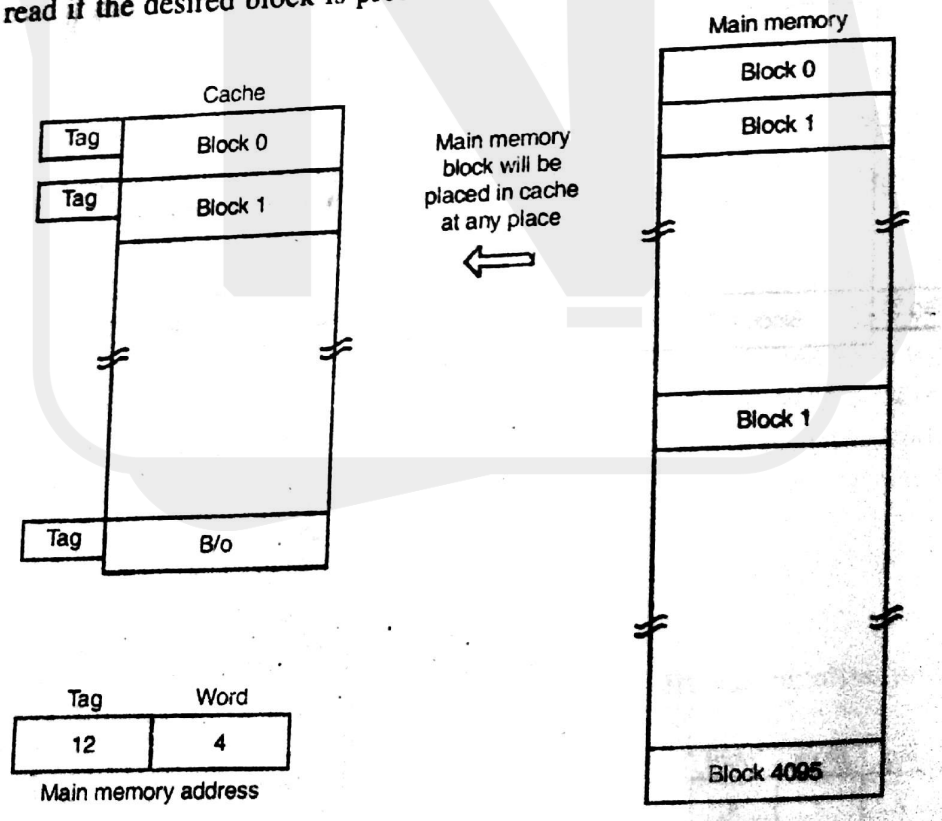

Fig. 9.17. Associative mapped cache.

The cache space can be used more efficiently. A new block that has to be brought into the cache has to replace the existing block only if the cache is full. In this case, we need replacement algorithm to

replace existing block in the cache. The cost of an associative cache is higher than the cost of direct mapping cache because of the need to search all 128 tag patterns to determine whether a given block is in the cache.

(c) Set Associative Mapping

Set associative mapping is a combination of both direct and associative mapping. It contains several groups of direct mapped blocks that operate in parallel. A block of data from any page in the main memory can go into a particular block of any direct-mapping cache. Hence the contention problem of the direct mapped technique is cased by having a few choices for block placement. The required address comparisons depend on the number of direct mapped cache in the cache system. These comparisons are always less than the comparisons required in the fully associative mapping.

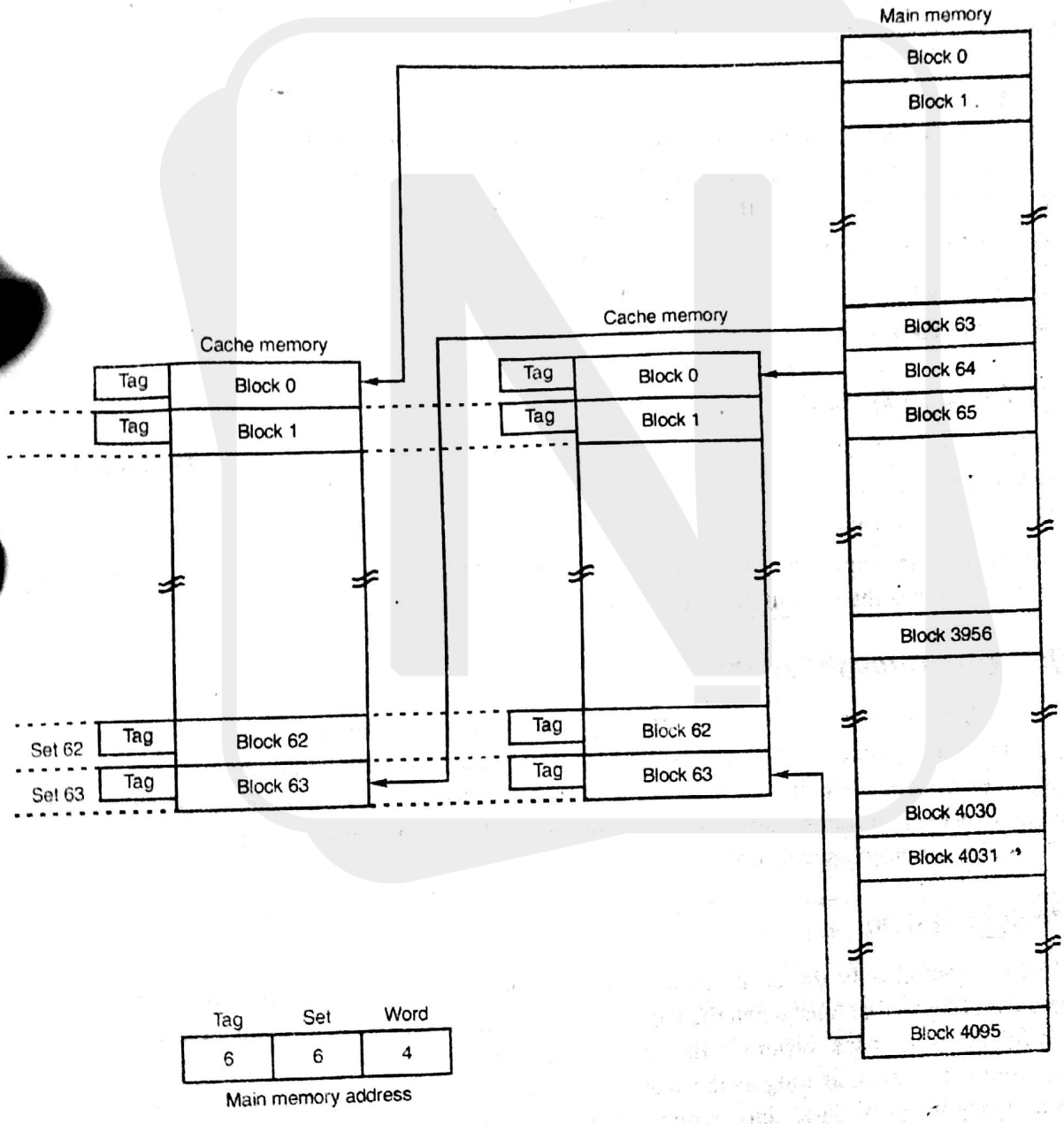

Fig. 9.18. Set asosciative cache.

Figure 9.18 shows the set asosciative mapped cache with two blocks per set. Each page in the main memory is organised in such a way that the size of each page is same as the size of one directly mapping

cahce. It is called two-way set associative cache because each block from main memory has two choice for block placement. In this technique, block 0, 64, 128, ..., 4032 of main memory can place into any of the two (block 0) blocks of set 0 and they occupy either of two block positions within this set. Having 64 sets means that the 6-bit set field of the address determines which set of the cache might contain the desired block. The tag field of the address must then be asosciatively compared to the tag of the two blocks of the set to check if the desired block is present. This two-way associative search is simple to implement.

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. The hit ratio is improved in this technique because the set size increase more words with same index but different tags can reside in cache. However, an increase in the set size increases the number of bits in words of cache and requires more complex comparison logic. 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 new data by using any one replacement algorithm.

(d) Cache Updat

9.4.4. Virtual Memory

In memory hierarchy systems, the physical main memory is not as large as the address space spanned by an address issued by CPU. When a program does not completely fit into the main memory the parts of

it (part of program or data) not currently being executed are stored on secondary storage devices such as magnetic disks and tapes.

If an executing program needs a segment which is not currently in the main memory, the required segment is copied from the secondary storage device. When a new segment of a program is to be copied into a main memory it must replace sanother segment already available in the memory.

In modern computers, the operating system moves program and data automatically between the main memory and secondary storage. Techniques that automatically swaps programmed data blocks between main memory and secondary storage device are called virtual memory. The addresses that CPU issues to access either instruction or data are called virtual or logical addresses. These addresses are translated into physical addresses by a combination of hardwares and software compoents. The virtual address refers to a part of the program or data space that is currently in the main memory, then the contents of the appropriate location in the main memory are accessed immediately. On the other hand, if the reference address is not in the main memory, its contents must be brought into a suitable location in the main memory before they can be used.

Figure 9.20 shows the typical implementation of virtual memory. A special hardware is Memory Management Unit (MMU) is used to translate the virtual address to physical addresses. When the desired data or programs are in the main memory, these data or program are fetched directly from the main memory. If the data are not in the main memory, the MMU comes the operating system to bring the data from the secondary storage like magnetic disk.

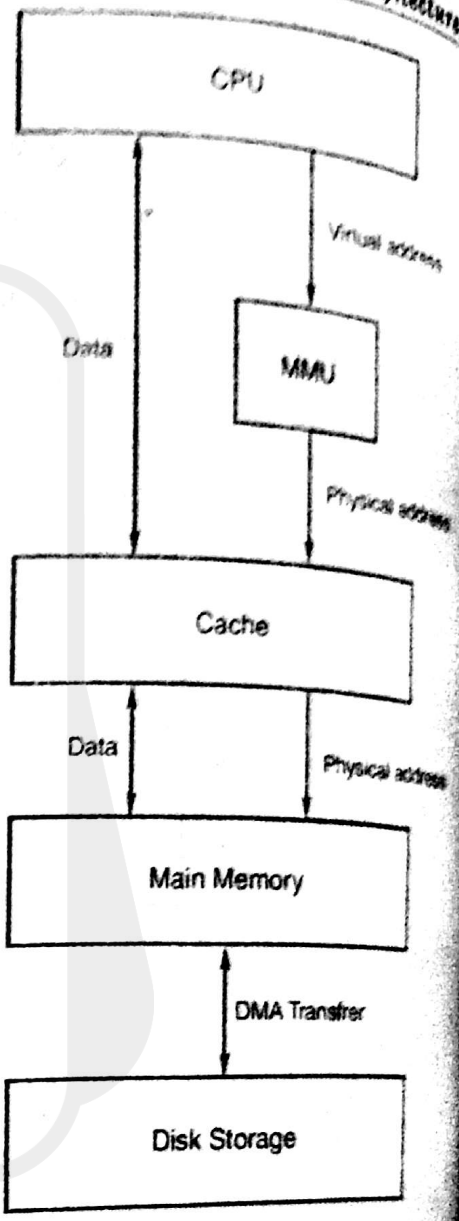

Fig. 9.20. Virtual memory organization.

Virtual Memory : A memory hierarchy consisting of main memory and auxiliary memory is known as virtual memory.

sed in a computer system is shown in Fig. 10.4.

Fig. 10.4. Printer.

10.2. INPUT-OUTPUT 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. A hardware unit that plays an important role between the CPU and peripheral devices to supervise the input and output transfers is known as interface.

The major differences are :

(1) Peripherals are electromechanical and electromagnetic devices and their manner of operations are different from the operation of the CPU and memory, which are electronic devices. Therefore, the conversion of signals are required.

(2) The data transfer rate of peripherals are usually slower than the transfer rate of the CPU and hence, a synchronization mechanism are needed.

(3) Data codes and formats in peripherals are different from the word format in the CPU and memory.

(4) The operating modes of peripherals are different from each other and each must be controlled so as not to disturb the operation of other peripherals connected to the CPU.

To resolve these differences, computer systems include special hardware known as "interface units" between the CPU and peripherals to supervise and synchronize all input and output transfers.

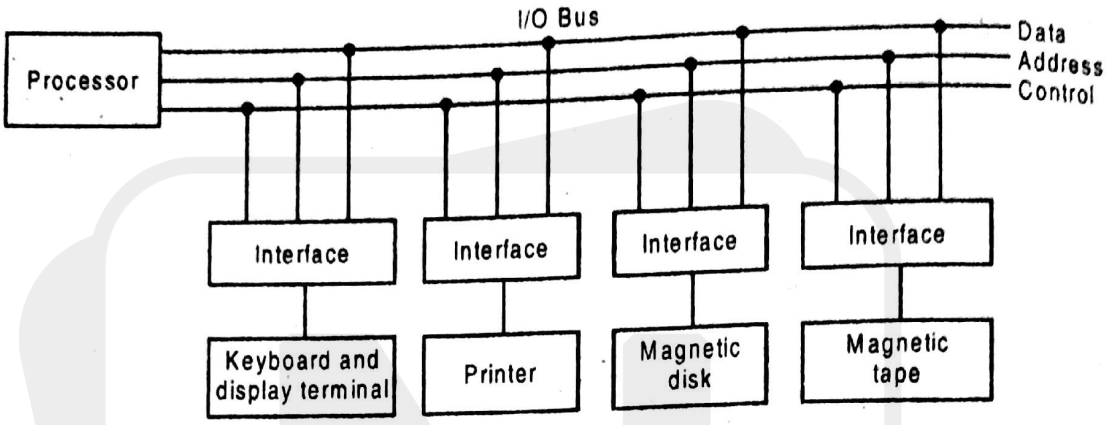

Fig. 10.5.

A typical communication link between the processor and several peripherals are shown in Fig. 10.5. The I/O bus consists of data lines, address lines and control lines.

Each peripheral devices has its associated interface unit. Each interface unit decodes the address and control received from 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 peripherals and processor. Each peripheral has its own controller that operates the particular electromechanical device. For example, the printer controller controls the paper motion, the printing timing and the selection of printing characters.

The I/O bus from the processor is attached to all peripherals 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 devices that it controls. All peripherals whose address does not correspond to the address in bus are disabled by their interfaces.

At the same time that the address is made available in the address lines, the processor provides a function code in the control lines. The interface selected responds to the function code and proceeds to execute it. These function codes are referred to as I/O commands. There are four types of commands that the interface may receive. They are :

(*i*) **Control Command :** A control command is issued to activate the peripheral and to inform it what to do. The particular control command issued depends on the peripheral.

For example, a magnetic tape unit may be instructed to back space the tape by one record, to rewind the tape or to start the tape moving in the forward direction.

(*ii*) **Status :** A status command is used to test various status conditions in the interface and the peripherals.

(*iii*) **Output Data :** A data output command causes the interface to respond by transferring data from the bus into one of its registers.

(*iv*) **Input Data :** 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 the

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.

10.2.1. I/O Versus Memory Bus

The processor must communicate to the memory unit along with the I/O unit. The memory bus also contains the data, address and readwrite control lines as in I/O bus. The computer buses can communicate to memory and I/O in three different ways :

(*i*) Use two separate buses, one of memory and one for I/O.

(*ii*) Use one common bus for both memory and I/O bus having separate control lines for each.

(*iii*) Use one common bus and common control lines for both memory and I/O.

10.2.2. Isolated Versus Memory-Mapped I/O

The transfer of information between the CPU and the memory or I/O are done by means of a common bus. By enabling the specific read or write lines, the CPU specifies whether the address on the address bus is for a memory word or for an interface registers. During a memory transfer, the memory read or memory write control lines are enabled and during I/O transfer, the I/O read or I/O write control lines are enabled. In this way the I/O interface addresses are isolated from the memory addresses and is termed as isolated I/O method for assigning addresses in a common bus.

In isolated I/O configuration, the input and output instructions for both interface register and memory are different. When the CPU fetches and decodes the instructions for I/O interfaces, it enables the I/O read or I/O write control line and informs the other components attached to the common bus that the address in the address bus is for interface register and not for memory word. Similarly, when the CPU fetches and decodes the instruction from memory, it places the memory address on the address bus and enables the memory read or memory write control line, which inform the external components that the address is for memory word and not for I/O interface. Hence, the memory and I/O addresses are isolated in this method.

Alternative, method is known as memory mapped I/O. In this method the computer use the same address space for memory and I/O, both. In memory-mapped I/O there is no specific input or output instructions. The same instruction can be used for manipulating I/O data residing in interfaces and can also be used to manipulate memory words. Computer with memory-mapped I/O can use memory-type instructions to access I/O data and hence, allows the computer to use the same instructions for memory transfers or input-output transfers.

3.7.5. Bus Arbitration

Amongst several masters and slave units are connected to a shared bus, it may happen that more than one master or slave units will request access to the bus at a same time. In such situations bus access is give to the master having highest priority. A mechanism which decides the selection of current master to access bus is known as bus arbitration. Three different mechanisms are commonly used for this.

1. Daisy chaining
2. Parallel arbitration
3. Independent requesting

IMPORTANT NOTES

- **Bus Arbitration:** A mechanism which decides the selection of current master to access bus is known as bus arbitration.

3.7.5.1. Daisy Chaining

The system connection for daisy chaining method are shown in Fig. 3.23. Daisy chaining method is cheaper and simple method. All masters make use of the same line for bus request. The bus grant signal serially propagates through each master until it encounters the first one that is requesting access to the bus. This master blocks the propagation of the bus grant signal, activates the busy line and gains control of the bus. Therefore, any other requesting module will not receive the grant signal and hence cannot get the bus access.

Advantages

1. It is a simple and cheapter method.
2. It requires the least number of lines and this number is independent of the number of masters in the system.

Disadvantages

1. The propagation time delay of bus grant signal is proportional to the number of masters in the system. This makes arbitration time slow and hence limits the number of masters in the system.
2. The priority of the master is fixed by the physical location of master.
3. Failure of any one master causes the whole system to fail.

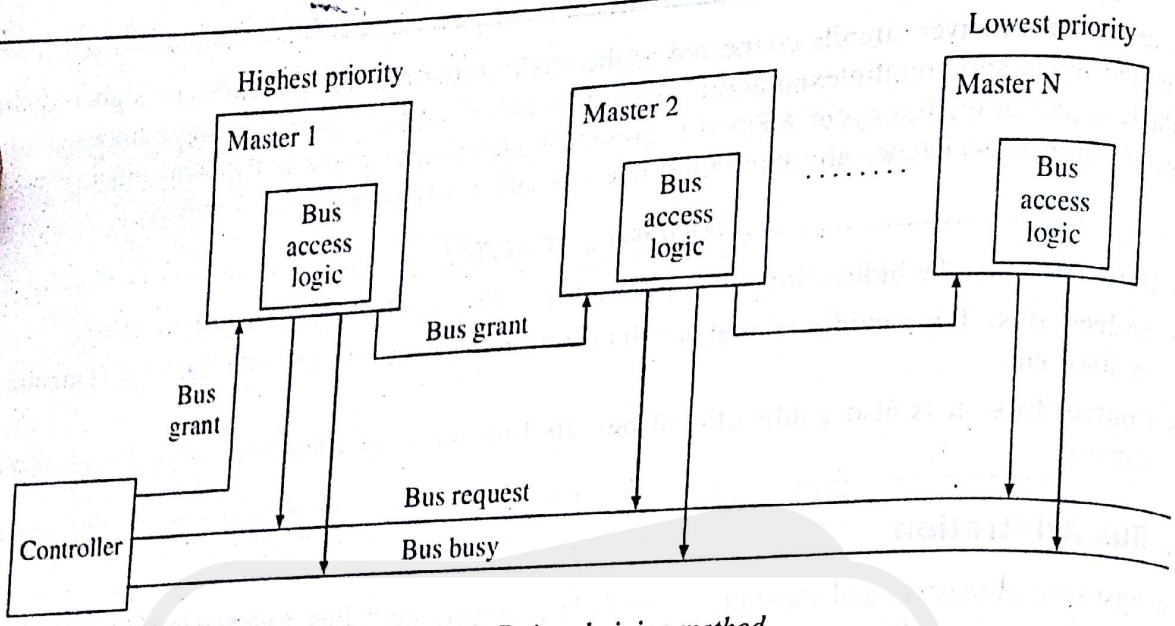

Fig. 3.19. Daisy chaining method

3.7.5.2. Parallel Arbitration

Figure 3.20 shows the parallel arbitration logic, it consists of priority encoder and a decoder. In this mechanism, each bus arbiter has a bus request output line and a bus acknowledges input line. Each arbiter enables the request line when its device (processor) is requesting access to the system bus.

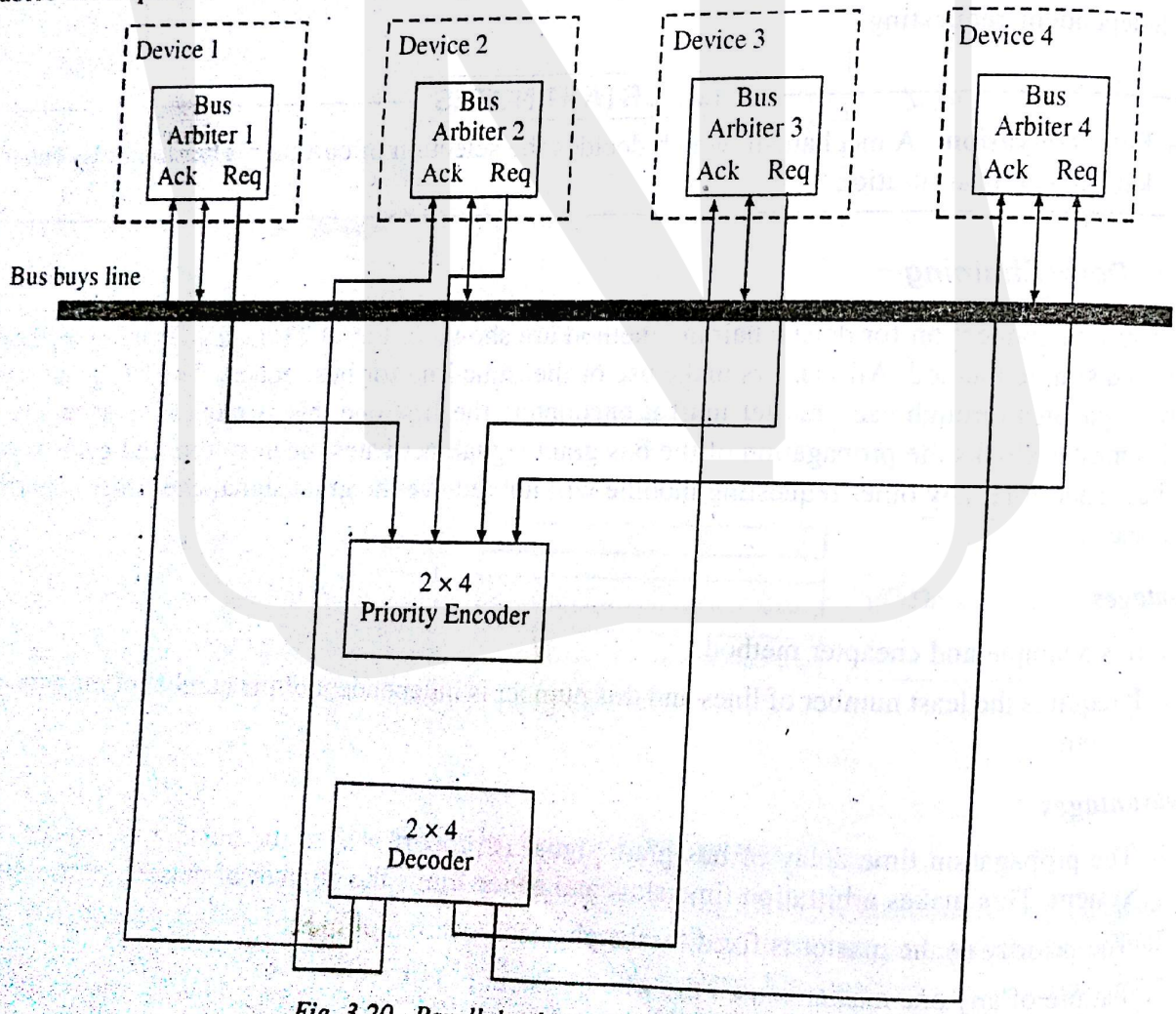

Fig. 3.20. Parallel arbiter for four devices.

Each arbiter bus request output line is connected to the input of priority encoder. The output of the encoder generates a 2-bit code which represents the highest-priority unit among those requesting the bus. This 2-bit code is given to the input of 2 × 4 decoder which enables the proper acknowledge line to grant bus access to the highest-priority unit. A device is utilised in the system only after it receives the acknowledged signal.

Advantages

1. The priority can be changed by altering the priority sequence stored in the controller.
2. If one module fails entire system does not fail.

3.7.5.3. Independent Priority

Figure 3.21 shows the system connections for the independent priority scheme. In this scheme each master has a separate pair of bus request and bus grant lines and each pair has a priority assigned to it. The built in priority encoder within the controller selects the highest priority and asserts the corresponding bus grant signal.

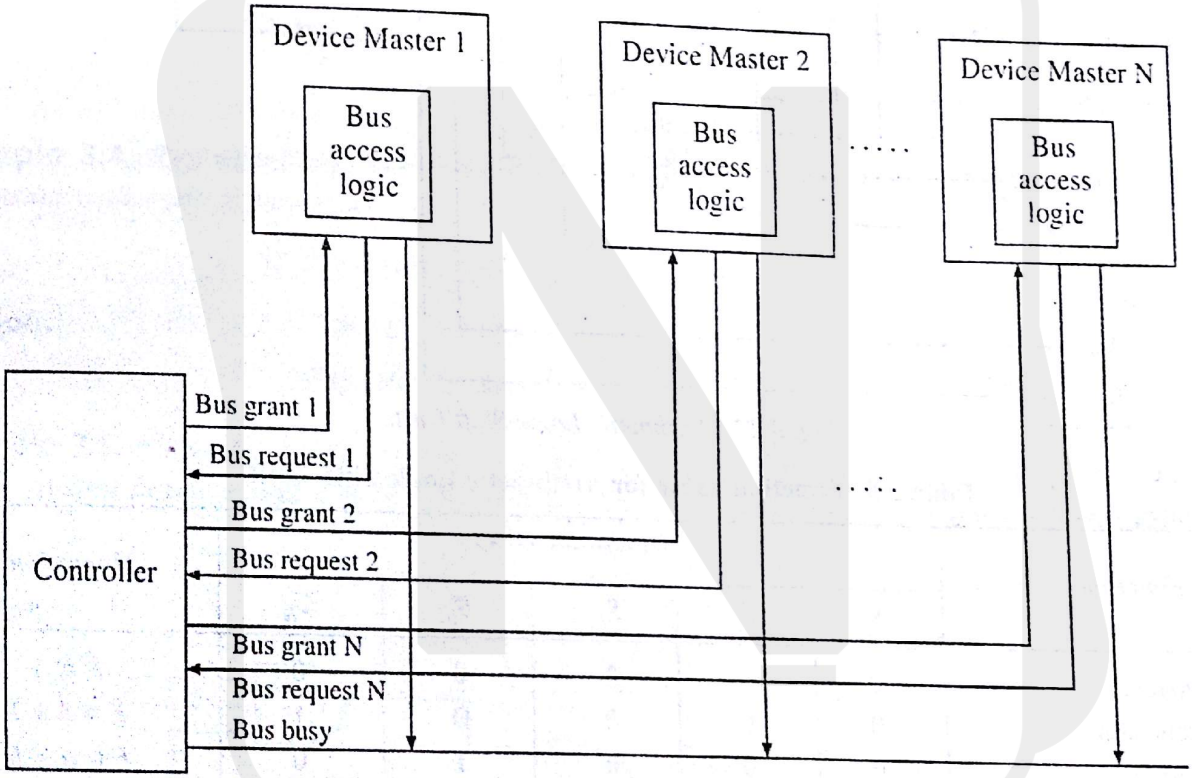

Fig. 3.21. Independent priority method.

3.8. DESIGN OF AN ARITHMETIC LOGIC SHIFT UNIT

In the earlier sections, we discussed the logic circuits for arithmetic operations and logic operations. These circuits can be combined to form one ALU with common selection variables. Figure 3.22 shows one stage of a complete arithmetic logic shift unit. The arithmetic, logic and shift operations are performed on the contents of two registers A and B. A_i stands for the ith bit of register A and B_i stands for the ith bit of register B. Inputs A_i and B_i are applied to both the arithmetic and shift units.

S_1 and S_0 selects a particular microoperation. A 4 × 1 multiplexer choose between the arithmetic output D_i and logic output E_i. The data input of the multiplexer are selected by the selection lines S_3

and S_2, A_{i-1} and A_{i+1} are the two data inputs of multiplexer for shift-right operation and shift-left operation, respectively. The output carry C_{i+1} is connected to the input carry C_i of the next stage in sequence.

Table 3.8 lists eight arithmetic operation, four logic operation and two shift operations. S_3, S_2, S_1, S_0 and C_{in} selects a particular operation. C_{in} is used to select the arithmetic operation only.

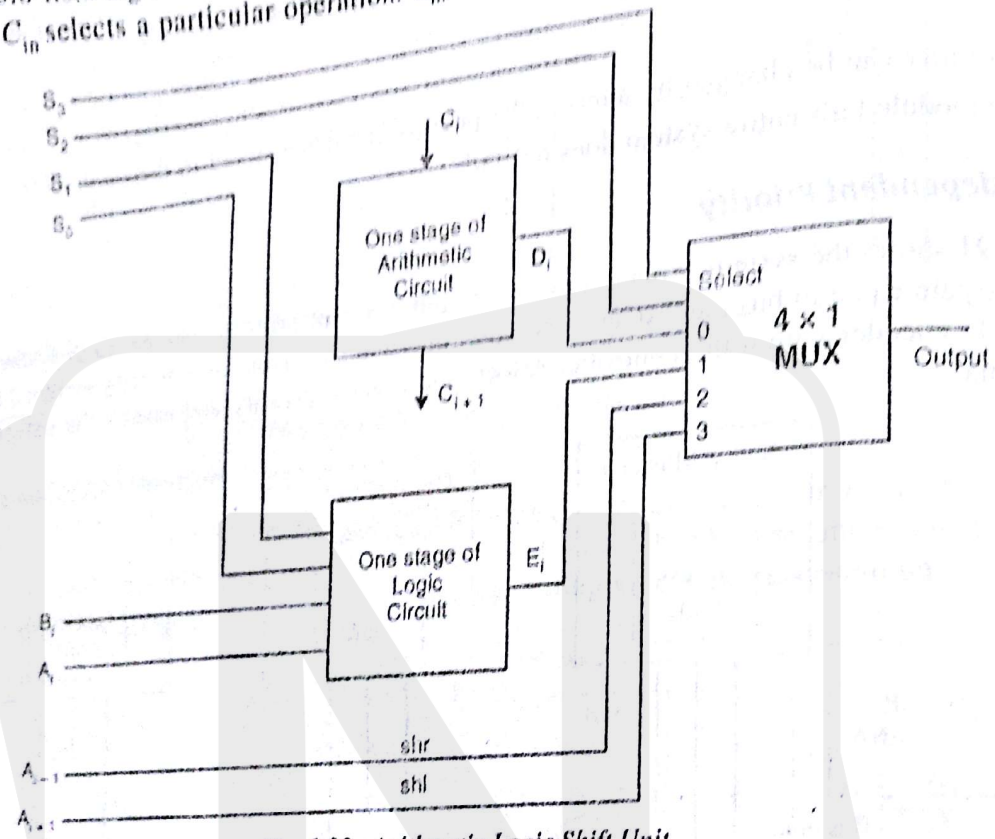

Fig. 3.22. Arithmetic Logic Shift Unit.

Table 3.8. Function Table for Arithmetic Logic Shift Unit.

Function	Operation select					Operation
	S_3	S_2	S_1	S_0	C_{in}	
Transfer A	0	0	0	0	0	$F = A$
Increment A	0	0	0	0	1	$F = A + 1$
Addition	0	0	0	1	0	$F = A + B$
Add with carry	0	0	0	1	1	$F = A + B + 1$
Subtract with borrow	0	0	1	0	0	$F = A + \overline{B}$
Subtraction	0	0	1	0	1	$F = A + \overline{B} + 1$
Decrement A	0	0	1	1	0	$F = A - 1$
Transfer A	0	0	1	1	1	$F = A$
AND	0	1	0	0	×	$F = A \wedge B$
OR	0	1	0	1	×	$F = A \vee B$
XOR	0	1	1	0	×	$F = A \oplus B$
Complement A	0	1	1	1	×	$F = \overline{A}$
Shift right A into F	1	0	×	×	×	$F = \text{shr } A$
Shift left A into F	1	1	×	×	×	$F = \text{shl } A$