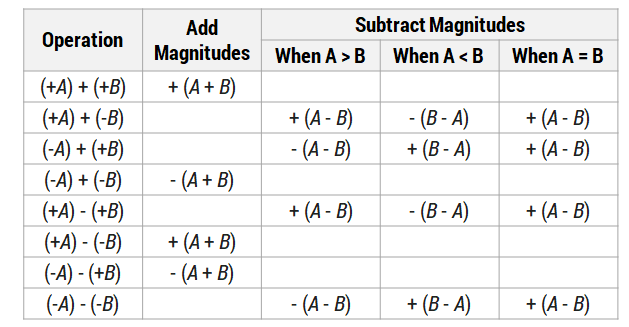
**UNIT-IV**

**Computer Arithmetic**

* + Addition and Subtraction
  + Multiplication Algorithms (Booth Multiplication Algorithm)

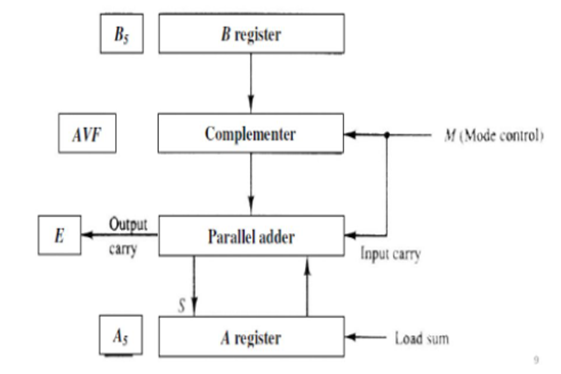
**Addition and Subtraction using signed magnitude data**

We designate the magnitude of the two numbers by A and B. Where the signed numbers  
are added or subtracted, we find that there are eight different conditions to consider, dependingon the sign of the numbers and the operation performed.

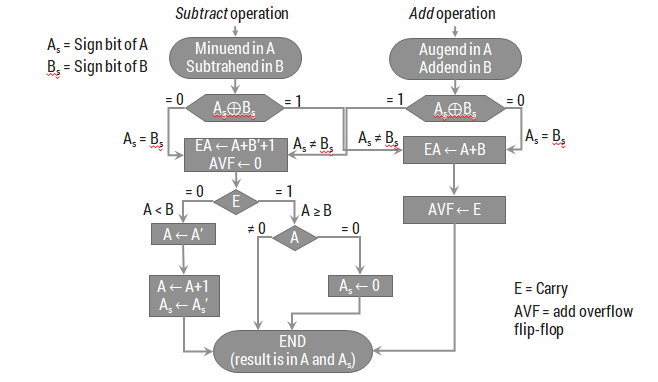
**Table: Addition and subtraction of sign-magnitude numbers**

When the signs of A and B are same, add the two magnitudes and attach the sign of result  
is that of A. When the signs of A and B are not same, compare the magnitudes and subtract thesmaller number from the larger. Choose the sign of the result to be the same as A, if A > B or thecomplement of the sign of A if A < B. If the two magnitudes are equal, subtract B from A andmake the sign of the result will be positive.

**Hardware Implementation**



**Flowchart for Addition & Subtraction**



**Example (-5)+(+6) consider 4 bits**

**-5 in sign magnitude : 1101 As=1 Augend(A)=101**

**+6 in sign magnitude :0110Bs=0 Addend(B)=110**

**As ex-or Bs=1 ex-or0 🡪0**

**EA🡪A+B``+1**

**EA🡪 101+ 010 🡪 111**

**E🡪0 A🡪111**

**AS E-🡪0 A🡪A```**

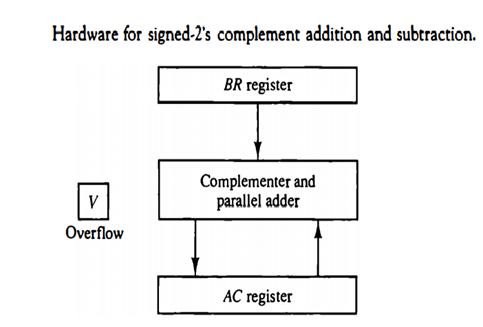
**A🡪000**

**A->A+1 = 001**

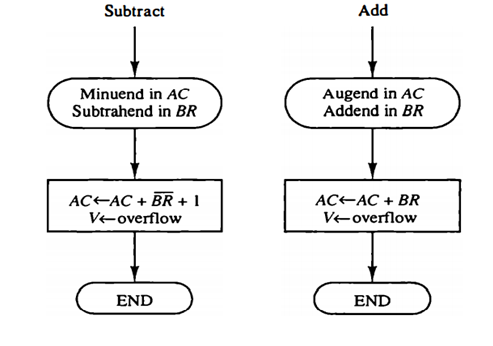
**AS🡪AS`` NOW AS=0**

**FINAL ANSWER IS 0001**

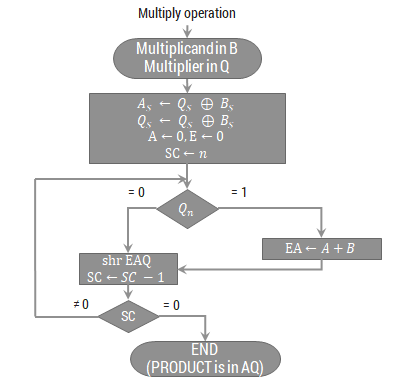
**Addition and subtraction using signed 2’s complement data**



**Algorithm for adding and subtracting numbers in signed 2's complement representation.**

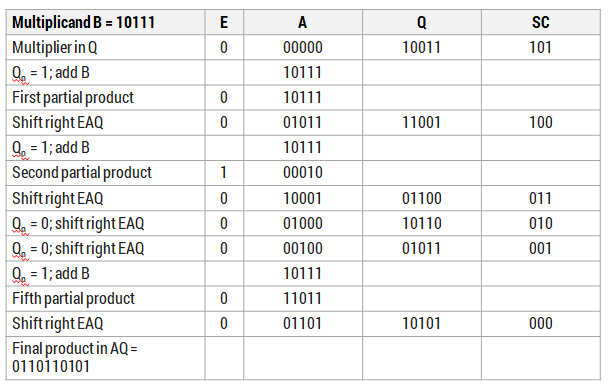


**Multiplication Algorithm for sign magnitude data**

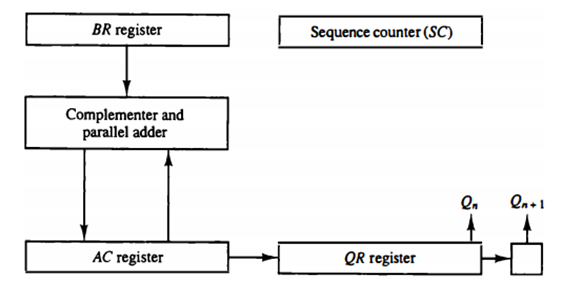


**Hardware Implementation**

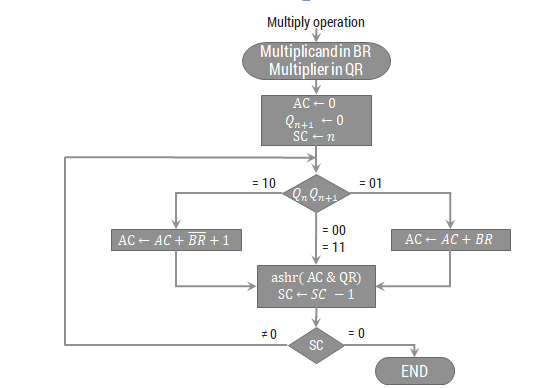
**Perform 23 x 19**



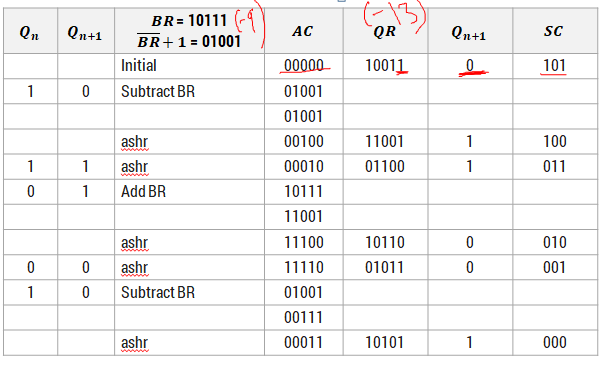
**Booth Multiplication Hardware(2’s complement multiplication)**



**Booth Multiplication Algorithm**



**Multiply (-9) x (-13) using Booth Algorithm**



**MEMORY ORGANISATION**

**Contents**

Memory Organization:

Memory Hierarchy,

Main Memory- RAM and ROM chips, Memory Address Map,

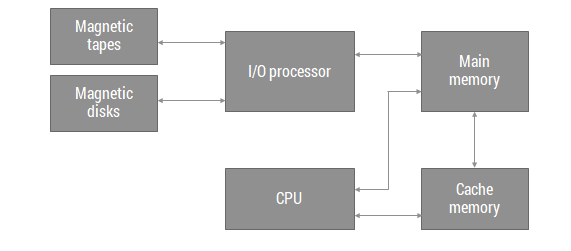
Auxiliary memory- Magnetic Disks, Magnetic Tapes,

Associative Memory – Hardware Organization,

Cache Memory – Associative Mapping, Direct Mapping, Set Associative Mapping,

Virtual Memory – Address Space and Memory Space, Address Mapping using pages, Associative Memory Page Table

**Memory Hierarchy**



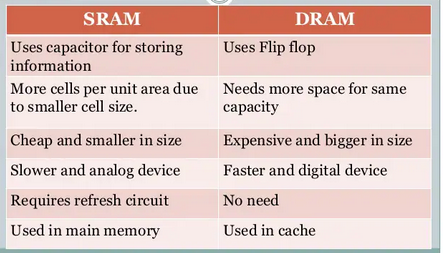
* The memory unit that communicates directly with the CPU is called main memory
* Devices that provide backup storage are called Auxilary memory .The most commonly used auxilary memory devices used in computer system are magnetic disks and tapes.
* They are used for storing system programs,large data files and other backup information
* Only the data and programs currently needed by the processor reside in the main memory
* All other information is stored in the auxilary memory and transferred to main memory when needed
* A special very high speed memory called Cache is sometimes used to increase the speed of processing
* The cache is used for storing segments of programs currently being executed by the CPU and temporary data frequently needed in present calculations

**Main Memory**

**Random access memory (RAM)**

* Used in computers for the temporary storage of programs and data.
* Read and write both operations are performed by RAM which requires fast cycle times as not to slow down the computer operation.
* It is volatile and lose all stored information if power is interrupted or turned off.
* It can be expanded by combining several memory chips.

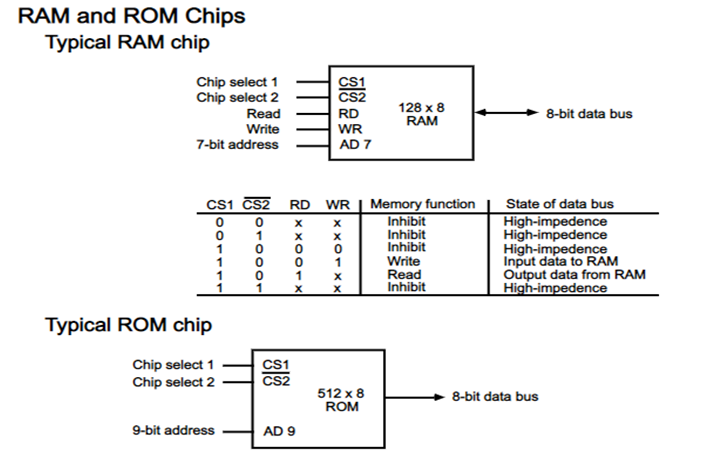
**SRAM VS DRAM**



**Read-Only Memory (ROM)**

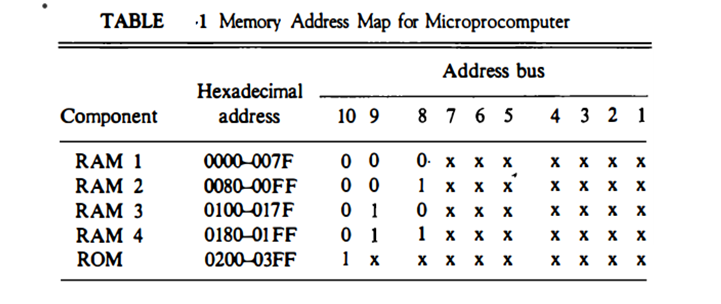
* A read-only memory (ROM) is essentially a memory device in which permanent binary information is stored.
* A ROM which can be programmed is called a PROM. The process of entering information in a ROM is known as programming.
* ROMs are used to store information which is of fixed type, such as tables for various functions, fixed data and instructions.

ROMs can be used for designing combinational logic circuits.



**Memory Address Map**

* The addressing of memory can establish by means of a table that specifies the memory address assigned to each chip.
* The table, called a memory address map, is a pictorial representation of assigned address space for each chip in the system, shown in the table.·
* To demonstrate with a particular example, assume that a computer system needs 512 bytes of RAM and 512 bytes of ROM.

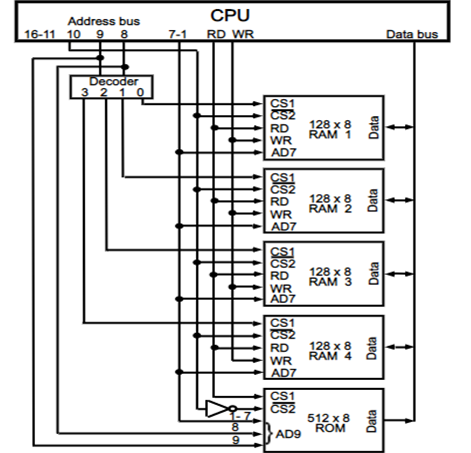


* The component column specifies whether a RAM or a ROM chip is used.
* The hexadecimal address column assigns a range of hexadecimal equivalent addresses for each chip.
* The address bus lines are listed in the third column.
* Although there are 16 lines in the address bus, the table shows only 10 lines because the other 6 are not used in this example and are assumed to be zero
* The small x's under the address bus lines designate those lines that must be connected to the address inputs in each chip.
* The RAM chips have 128 bytes and need seven address lines. The ROM chip has 512 bytes and needs 9 address lines.
* The x's are always assigned to the low-order bus lines: lines 1 through 7 for the RAM and lines 1 through 9 for the ROM.
* It is now necessary to distinguish between four RAM chips by assigning to each a different address.For this particular example we choose bus lines 8 and 9 to represent four distinct binary combinations.
* The table clearly shows that the nine low-order bus lines constitute a memory space for RAM equal to 29 = 512 bytes.
* The distinction between a RAM and ROM address is done with another bus line. Here we choose line 10 for this purpose.

When line 10 is 0, the CPU selects a RAM, and when this line is equal to 1, it selects the ROM

**Memory Connection to CPU**

* RAM and ROM Chips are connected to a CPU through the data and address buses



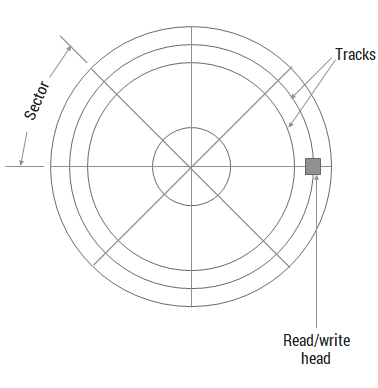
**Auxiliary Memory**

* Most common magnetic disks and tapes
* Other-Magnetic drums and optical disks
* To understand fully the physical mechanism of auxiliary memory devices one must have a knowledge of magnetics, electronics, and electromechanical system
* The important characteristics of any device are its access mode, access time, transfer rate, capacity, and cost.
* The average time required to reach a storage location in memory and obtain its contents is called the access time
* Access time=seektime+transfertime
* Seektime=to position the read-write head of a location
* Transfer time= time to transfer data

Transfer rate-number of characters or words that the device can transfer per second

**Magnetic Disks**

* Circular plate of metal or plastic coated with magnetized material.
* Often both sides of the disk are used and several disks may be stacked on one spindle with read/write heads available on each surface.
* All disks rotate together at high speed and are not stopped or started for access purposes.
* Bits are stored in the magnetized surface in spots along concentric circles called track.
* The tracks are commonly divided into sections called sectors.
* A disk system is addressed by address bits that specify the disk number, the disk surface, the sector number and the track within the sector.
* Disks that are permanently attached and cannot be removed by an occasional user are called hard disks

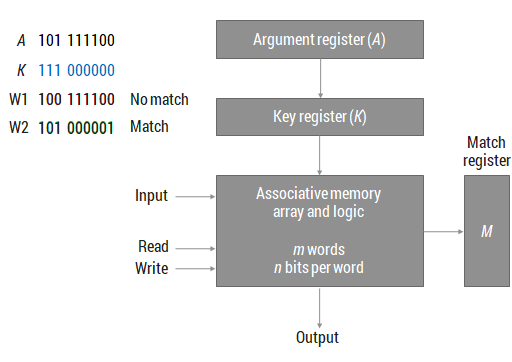


**Magnetic Tape**

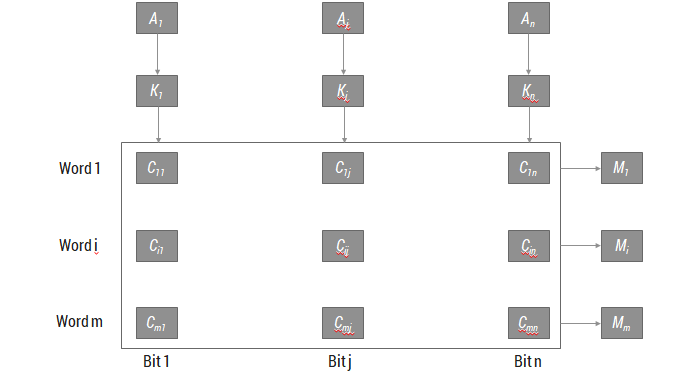
* Magnetic tape transport consists of the electrical, mechanical and electronic components to provide the parts and control mechanism for a magnetic-tape unit.
* The tape itself is a strip of plastic coated with a magnetic recording medium.
* Bits are recorded as magnetic spots on the tape along several tracks called records
* Magnetic tape units can be stopped, started to move forward or in reverse, or can be rewound. However, they cannot be started or stopped fast enough between individual characters.
* A tape unit is addressed by specifying the record number and the number of characters in the record.

**Associative Memory(Content Addressable Memory)**

* The time required to find an item stored in memory can be reduced considerably if stored data can be identified for access by the content of the data itself rather than by an address.
* A memory unit accessed by content is called an associative memory or content addressable memory (CAM).
* This type of memory is accessed simultaneously and in parallel on the basis of data content rather than by specific address or location.
* When a word is to be read from an associative memory, then the content of the word or part of word is specified



**Associative Memory of m words n cells per word**



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

**Cache Memory**

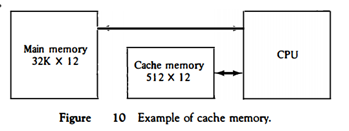
* Cache is a fast small capacity memory that should hold those information which are most likely to be accessed.
* **Analysis** of a large number of typical programs has shown that the references to memory at any given interval of time tend to be confined within a few localized areas in memory.

**This phenomenon** is known as the property of locality of reference locality of reference.

* 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**

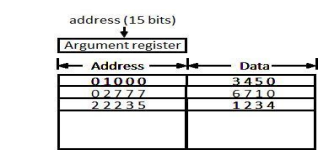
* The performance of the cache memory is frequently measured in terms of a quantity called ***hit ratio***.
* When the CPU refers to memory and finds the word in cache, it is said to produce a ***hit*.**
* If the word is not found in cache, it is in main memory and it counts as a ***miss.***
* The ratio of the number of hits divided by the total CPU references to memory (hits plus misses) is the *hit ratio.*
* Hit ratios of 0.9 and higher have been reported.



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

**Cache Mapping Techniques**

* 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.



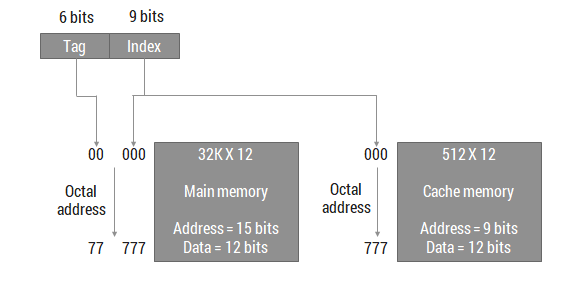
* The associative memory stores both the address and content (data) of the memory word. This permits any location in cache to store any word from main memory.
* The above figure shows three words presently stored in the cache
* The address value of 15 bits is shown as a five-digit octal number and its 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.

**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 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

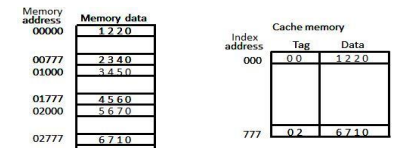
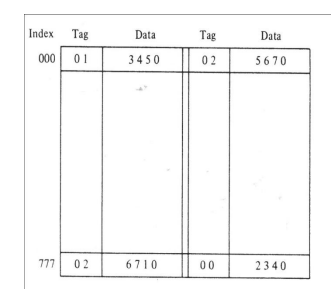


Fig:Direct mapping cache organisation

* 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 thecache.
* 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 butwith different tag values cannot reside in cache memory at the same time.

.**Set-Associative Mapping**

****

A third type of cache organization, called set-associative mapping, is an improvement over the direct-mapping organization in that each word of cache can store two or more words of memory under the same index address. Each data word is stored together with its tag and the number of tag data items in one word of cache is said to form a set. Each tag requires six bits and each data word has 12 bits, so the word length is (6 + 12) = 36 bits. An index address of nine bits can accommodate 512 words. Thus the size of cache memory is 512 × 36.

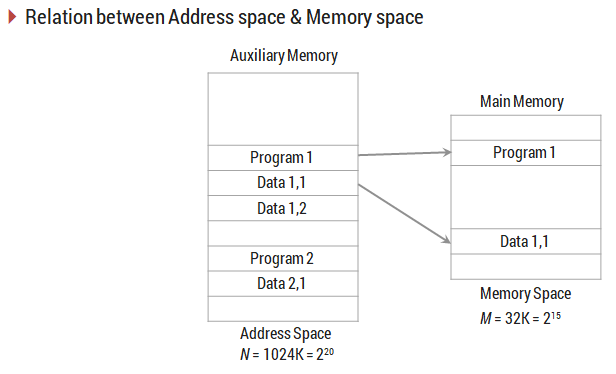
The words stored at addresses 01000 and 02000 of main memory are stored in cache memory at index address 000. Similarly, the words at addresses 02777 and 00777 are stored in cache at index address 777. When the CPU generates a memory request, the index value of the address is used to access the cache. The tag field of the CPU address is then compared with both tags in the cache to determine if a match occurs

**Virtual Memory**

Virtual memory is used to give programmers the illusion that they have a very large memory even though the computer has a small main memory. It makes the task of programming easier because the programmer no longer needs to worry about the amount of physical memory available.

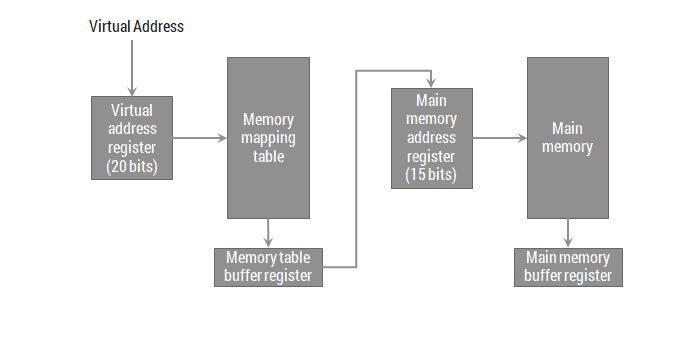
**Address space**An address used by a programmer will be called a virtual address, and the set of such addresses is known as address space.

An address in main memory is called a location or physical address. The set of such locations is called the **memory space**

****

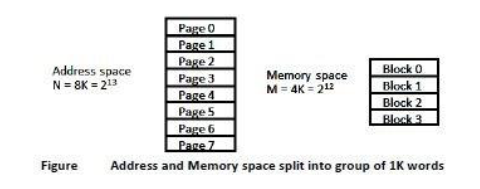
**Memory table for mapping a virtual address**

* **A table** is then needed, as shown in Fig, to map a virtual address of 20 bits to a physical address of 15 bits.
* **The mapping table** may be stored in a separate memory as shown in Fig or in main memory. In the first case, an additional memory unit is required as well as one extra memory access time.

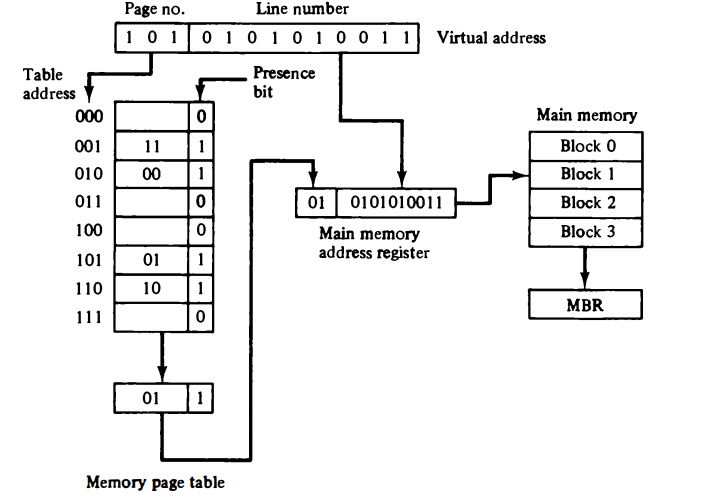


**Address mapping using pages**

* The table implementation of the address mapping is simplified if the information in the address space. And the memory space is each divided into groups of fixed size.
* Moreover, The physical memory is broken down into groups of equal size called blocks, which may range from 64 to 4096 words each.
* The term page refers to groups of address space of the same size.
* Also, Consider a computer with an address space of 8K and a memory space of 4K.
* If we split each into groups of 1K words we obtain eight pages and four blocks as shown in the figure.
* At any given time, up to four pages of address space may reside in main memory in any one of the four blocks.



**Memory Table in a paged system**



* **The memory-page table** consists of eight words, one for each page. The address in the page table denotes the page number and the content of the word gives the block number where that page is stored in main memory.

* **The table shows** that pages 1, 2, 5, and 6 are now available in main memory in blocks 3, 0, 1, and 2, respectively.

* **A presence bit** in each location indicates whether the page has been transferred from auxiliary memory into main memory. A 0 in the presence bit indicates that this page is not available in main memory.

* **The CPU references** a word in memory with a virtual address of 13 bits. The three high-order bits of the virtual address specify a page number and also an address for the memory-page table.
* **The content of the word** in the memory page table at the page number address is read out into the memory table buffer register.

* **If the presence bit** is a 1, the block number thus read is transferred to the two high-order bits of the main memory address register.

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

**Associative Memory Page Table**

* A random-access memory page table is inefficient with respect to storage utilization.
* consider an address space of 1024K words and memory space of 32K words. If each page or block contains 1K words, the number of pages is 1024 and the number of blocks 32. The capacity of the memory-page table must be 1024 words and only 32 locations may have a presence bit equal to 1. At any given time, at least 992 locations will be empty and not in use.
* A more efficient way to organize the page table would be to construct it with a number of words equal to the number of blocks in main memory.

