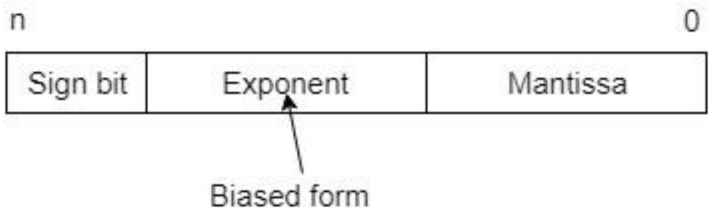
UNIT-III

Floating-Point Representation −

This representation does not reserve a specific number of bits for the integer part or the fractional part. Instead it reserves a certain number of bits for the number (called the mantissa or significand) and a certain number of bits to say where within that number the decimal place sits (called the exponent).

The floating number representation of a number has two part: the first part represents a signed fixed point number called mantissa. The second part of designates the position of the decimal (or binary) point and is called the exponent. The fixed point mantissa may be fraction or an integer. Floating -point is always interpreted to represent a number in the following form: Mxr^e.

Only the mantissa m and the exponent e are physically represented in the register (including their sign). A floating-point binary number is represented in a similar manner except that is uses base 2 for the exponent. A floating-point number is said to be normalized if the most significant digit of the mantissa is 1.



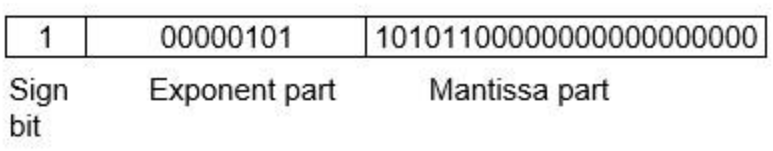
So, actual number is $(-1)^s(1+m)x2^{(e\text{-}Bias)}$, where *s* is the sign bit, *m* is the mantissa, *e* is the exponent value, and *Bias* is the bias number.

Note that signed integers and exponent are represented by either sign representation, or one's complement representation, or two's complement representation.

The floating point representation is more flexible. Any non-zero number can be represented in the normalized form of $\pm(1.b_1b_2b_3\ ...)_2x2^n$ This is normalized form of a number x.

Example −Suppose number is using 32-bit format: the 1 bit sign bit, 8 bits for signed exponent, and 23 bits for the fractional part. The leading bit 1 is not stored (as it is always 1 for a normalized number) and is referred to as a *"hidden bit*".

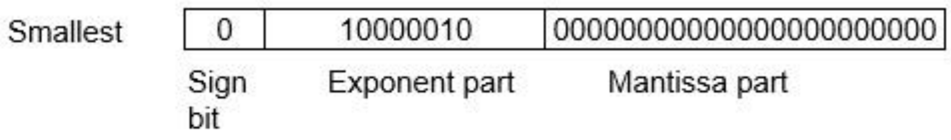
Then −53.5 is normalized as $-53.5=(-110101.1)_2=(-1.101011)x2^5$, which is represented as following below,

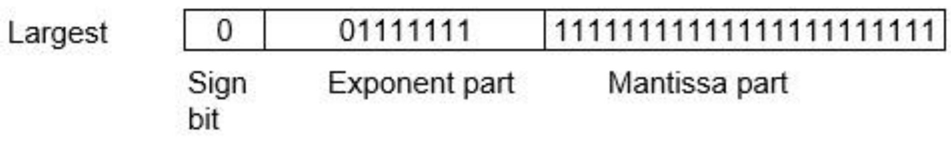


Where 00000101 is the 8-bit binary value of exponent value +5.

Note that 8-bit exponent field is used to store integer exponents $-126 \leq n \leq 127$.

The smallest normalized positive number that fits into 32 bits is $(1.00000000000000000000000)_2 x 2^{-126} = 2^{-126} \approx 1.18x10^{-38}$, and largest normalized positive number that fits into 32 bits is $(1.11111111111111111111111)_2 x 2^{127} = (2^{24}-1) x 2^{104} \approx 3.40x10^{38}$. These numbers are represented as following below,





Unsigned and Signed Binary Numbers

Data Structure and AlgorithmsMathematicsDigital Electronics

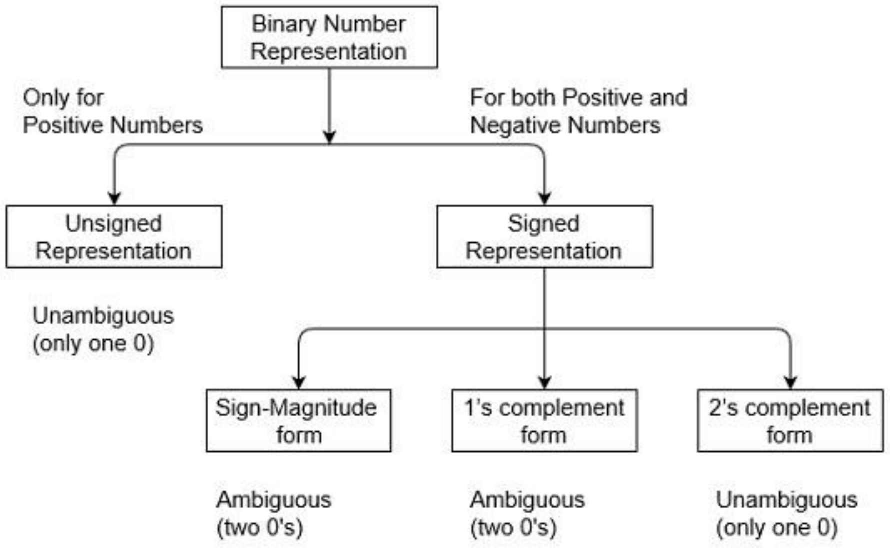
Variables such as integers can be represent in two ways, i.e., signed and unsigned. Signed numbers use sign flag or can be distinguish between negative values and positive values. Whereas unsigned numbers stored only positive numbers but not negative numbers.

Number representation techniques like: **Binary**, **Octal**, **Decimal**, and **Hexadecimal** number representation techniques can represent numbers in both signed and unsigned ways. Binary Number System is one the type of Number Representation techniques. It is most popular and used in digital systems. Binary system is used for representing binary quantities which can be represented by any device that has only two operating states or possible conditions. For example, a switch has only two states: open or close.

In the Binary System, there are only two symbols or possible digit values, i.e., 0 and 1. Represented by any device that only 2 operating states or possible conditions. Binary numbers are indicated by the addition of either an *0b* prefix or an *2* suffix.

Representation of Binary Numbers:

Binary numbers can be represented in signed and unsigned way. Unsigned binary numbers do not have sign bit, whereas signed binary numbers uses signed bit as well or these can be distinguishable between positive and negative numbers. A signed binary is a specific data type of a signed variable.



1. Unsigned Numbers:

Unsigned numbers don't have any sign, these can contain only magnitude of the number. So, representation of unsigned binary numbers are all positive numbers only. For example, representation of positive decimal numbers are positive by default. We always assume that there is a positive sign symbol in front of every number.

Representation of Unsigned Binary Numbers:

Since there is no sign bit in this unsigned binary number, so N bit binary number represent its magnitude only. Zero (0) is also unsigned number. This representation has only one zero (0), which is always positive. Every number in unsigned number representation has only one unique binary equivalent form, so this is unambiguous representation technique. The range of unsigned binary number is from 0 to (2^n-1).

Example-1: Represent decimal number 92 in unsigned binary number.

Simply convert it into Binary number, it contains only magnitude of the given number. = $(92)_{10}$

= $(1x2^6+0x2^5+1x2^4+1x2^3+1x2^2+0x2^1+0x2^0)_{10}$

= $(1011100)_2$

It’s 7 bit binary magnitude of the decimal number 92.

Example-2: Find range of 5 bit unsigned binary numbers. Also, find minimum and maximum value in this range.

Since, range of unsigned binary number is from 0 to (2^n-1). Therefore, range of 5 bit unsigned binary number is *from* 0 to (2^5-1) which is equal from minimum value 0 (i.e., 00000) to maximum value 31 (i.e., 11111).

2. Signed Numbers:

Signed numbers contain sign flag, this representation distinguish positive and negative numbers. This technique contains both sign bit and magnitude of a number. For example, in representation of negative decimal numbers, we need to put negative symbol in front of given decimal number.

Representation of Signed Binary Numbers:

There are three types of representations for signed binary numbers. Because of extra signed bit, binary number zero has two representation, either positive (0) or negative (1), so ambiguous representation. But 2’s complementation representation is unambiguous representation because of there is no double representation of number 0. These are: Sign-Magnitude form, 1’s complement form, and 2’s complement form which are explained as following below.

2.(a) Sign-Magnitude form:

For n bit binary number, 1 bit is reserved for sign symbol. If the value of sign bit is 0, then the given number will be positive, else if the value of sign bit is 1, then the given number will be negative. Remaining (n-1) bits represent magnitude of the number. Since magnitude of number zero (0) is always 0, so there can be two representation of number zero (0), positive (+0) and negative (-0), which depends on value of sign bit. Hence these representations are ambiguous generally because of two representation of number zero (0). Generally sign bit is a most significant bit (MSB) of representation. The range of Sign-Magnitude form is from ($2^{(n-1)}$-1) to ($2^{(n-1)}$-1).

For example, range of 6 bit Sign-Magnitude form binary number is from (2^5-1) to (2^5-1) which is equal from minimum value -31 (i.e., 1 11111) to maximum value +31 (i.e., 0 11111). And zero (0) has two representation, -0 (i.e., 1 00000) and +0 (i.e., 0 00000).

2.(b) 1’s complement form:

Since, 1’s complement of a number is obtained by inverting each bit of given number. So, we represent positive numbers in binary form and negative numbers in 1’s complement form. There is extra bit for sign representation. If value of sign bit is 0, then number is positive and you can directly represent it in simple binary form, but if value of sign bit 1, then number is negative and you have to take 1’s complement of given binary number. You can get negative number by 1’s complement of a positive number and positive number by using 1’s complement of a negative number. Therefore, in this representation, zero (0) can have two representation, that’s why 1’s complement form is also ambiguous form. The range of 1’s complement form is *from* $(2^{(n-1)}-1)$ to $(2^{(n-1)}-1)$.

For example, range of 6 bit 1’s complement form binary number is from (2^5-1) to (2^5-1) which is equal from minimum value -31 (i.e., 1 00000) to maximum value +31 (i.e., 0 11111). And zero (0) has two representation, -0 (i.e., 1 11111) and +0 (i.e., 0 00000).

2.(c) 2’s complement form:

Since, 2’s complement of a number is obtained by inverting each bit of given number plus 1 to least significant bit (LSB). So, we represent positive numbers in binary form and negative numbers in 2’s complement form. There is extra bit for sign representation. If value of sign bit is 0, then number is positive and you can directly represent it in simple binary form, but if value of sign bit 1, then number is negative and you have to take 2’s complement of given binary number. You can get negative number by 2’s complement of a positive number and positive number by directly using simple binary representation. If value of most significant bit (MSB) is 1, then take 2’s complement from, else not. Therefore, in this representation, zero (0) has only one (unique) representation which is always positive. The range of 2’s complement form is *from* $(2^{(n-1)})$ to $(2^{(n-1)}-1)$.

For example, range of 6 bit 2’s complement form binary number is from (2^5) to (2^5-1) which is equal from minimum value -32 (i.e., 1 00000) to maximum value +31 (i.e., 0 11111). And zero (0) has two representation, -0 (i.e., 1 11111) and +0 (i.e., 0 00000).

Computer Organization | Booth’s Algorithm

Last Updated : 27 Sep, 2024

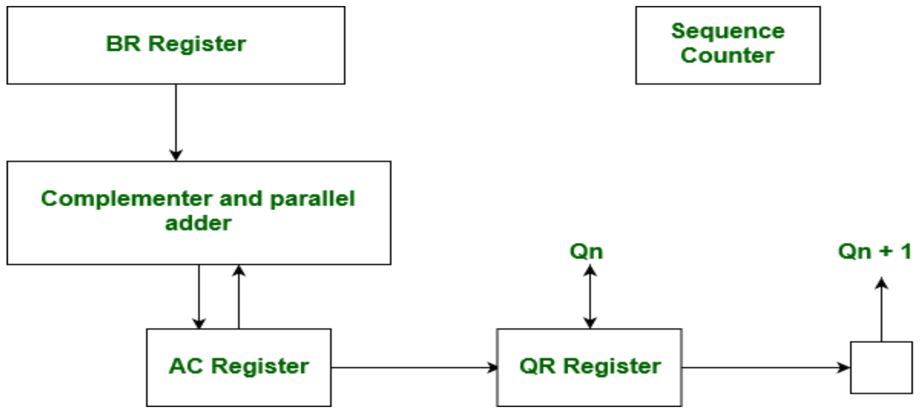
-

  Booth algorithm gives a procedure for **multiplying binary integers** in signed 2’s complement representation **in efficient way**, i.e., less number of additions/subtractions required. It operates on the fact that strings of 0’s in the multiplier require no addition but just shifting and a string of 1’s in the multiplier from bit weight 2^k to weight 2^m can be treated as 2^(k+1) to 2^m. As in all multiplication schemes, booth algorithm requires examination **of the multiplier bits** and shifting of the partial product. Prior to the shifting, the multiplicand may be added to the partial product, subtracted from the partial product, or left unchanged according to following rules:

1. The multiplicand is subtracted from the partial product upon encountering the first least significant 1 in a string of 1’s in the multiplier
2. The multiplicand is added to the partial product upon encountering the first 0 (provided that there was a previous ‘1’) in a string of 0’s in the multiplier.
3. The partial product does not change when the multiplier bit is identical to the previous multiplier bit.

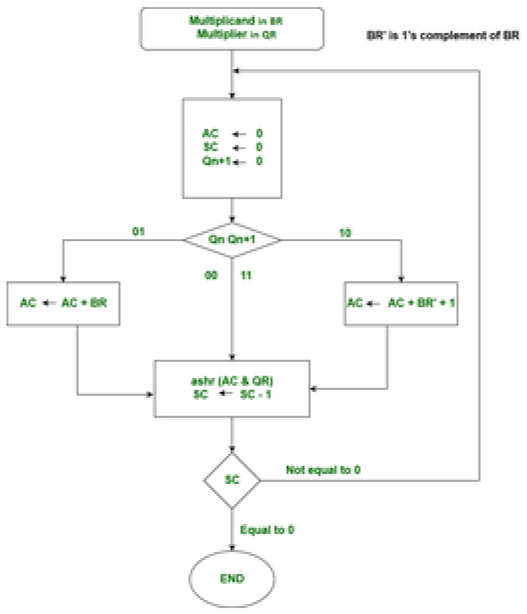
Booth’s Algorithm optimizes binary multiplication. For a better grasp of computer organization, the **GATE CS Self-Paced Course** provides clear explanations and examples.

Hardware Implementation of Booths Algorithm – The hardware implementation of the booth algorithm requires the register configuration shown in the figure below.



Booth’s Algorithm Flowchart –

We name the register as A, B and Q, AC, BR and QR respectively. Qn designates the least significant bit of multiplier in the register QR. An extra flip-flop Qn+1is appended to QR to facilitate a double inspection of the multiplier.The flowchart for the booth algorithm is shown below.



Flow chart of Booth’s Algorithm.

AC and the appended bit Qn+1 are initially cleared to 0 and the sequence SC is set to a number n equal to the number of bits in the multiplier. The two bits of the multiplier in Qn and Qn+1are inspected. If the two bits are equal to 10, it means that the first 1 in a string has been encountered. This requires subtraction of the multiplicand from the partial product in AC. If the 2 bits are equal to 01, it means that the first 0 in a string of 0’s has been encountered. This requires the addition of the multiplicand to the partial product in AC. When the two bits are equal, the partial product does not change. An overflow cannot occur because the addition and subtraction of the multiplicand follow each other. As a consequence, the 2 numbers that are added always have a opposite signs, a condition that excludes an overflow. The next step is to shift right the partial product and the multiplier (including Qn+1). This is an arithmetic shift right (ashr) operation which AC and QR to the right and leaves the sign bit in AC unchanged. The sequence counter is decremented and the computational loop is repeated n times. Product of negative numbers is important, while multiplying negative numbers we need to find 2's complement of the number to change its sign, because it's easier to add instead of performing binary subtraction. product of two negative number is demonstrated below along with 2's complement.

Advantages:

Faster than traditional multiplication: Booth's algorithm is faster than traditional multiplication methods, requiring fewer steps to produce the same result.

Efficient for signed numbers: The algorithm is designed specifically for multiplying signed binary numbers, making it a more efficient method for multiplication of signed numbers than traditional methods.

Lower hardware requirement: The algorithm requires fewer hardware resources than traditional multiplication methods, making it more suitable for applications with limited hardware resources.

Widely used in hardware: Booth's algorithm is widely used in hardware implementations of multiplication operations, including digital signal processors, microprocessors, and FPGAs.

Disadvantages:

Complex to understand: The algorithm is more complex to understand and implement than traditional multiplication methods.

Limited applicability: The algorithm is only applicable for multiplication of signed binary numbers, and cannot be used for multiplication of unsigned numbers or numbers in other formats without additional modifications.

Higher latency: The algorithm requires multiple iterations to calculate the result of a single multiplication operation, which increases the latency or delay in the calculation of the result.

Higher power consumption: The algorithm consumes more power compared to traditional multiplication methods, especially for larger inputs.

Application of Booth's Algorithm:

1. Chip and computer processors: Corner's Calculation is utilized in the equipment execution of number-crunching rationale units (ALUs) inside microchips and computer chips. These parts are liable for performing number juggling and coherent procedure on twofold information. Proficient duplication is fundamental in different applications, including logical registering, designs handling, and cryptography. Corner's Calculation lessens the quantity of piece movements and augmentations expected to perform duplication, bringing about quicker execution and better in general execution.

2. Digital Signal Processing (DSP): DSP applications frequently include complex numerical tasks, for example, sifting and convolution. Duplicating enormous twofold numbers is a principal activity in these errands. Corner's Calculation permits DSP frameworks to perform duplications all the more productively, empowering ongoing handling of sound, video, and different sorts of signs.

3. Hardware Accelerators: Many particular equipment gas pedals are intended to perform explicit assignments more productively than broadly useful processors. Corner's Calculation can be integrated into these gas pedals to accelerate augmentation activities in applications like picture handling, brain organizations, and AI.

4. Cryptography: Cryptographic calculations, like those utilized in encryption and computerized marks, frequently include particular exponentiation, which requires proficient duplication of huge numbers. Corner's Calculation can be utilized to speed up the measured augmentation step in these calculations, working on the general proficiency of cryptographic tasks.

5. High-Performance Computing (HPC): In logical reenactments and mathematical calculations, enormous scope augmentations are oftentimes experienced. Corner's Calculation can be carried out in equipment or programming to advance these duplication tasks and improve the general exhibition of HPC frameworks.
6. Implanted Frameworks: Inserted frameworks frequently have restricted assets regarding handling power and memory. By utilizing Corner's Calculation, fashioners can upgrade augmentation activities in these frameworks, permitting them to perform all the more proficiently while consuming less energy.
7. Network Parcel Handling: Organization gadgets and switches frequently need to perform estimations on bundle headers and payloads. Augmentation activities are regularly utilized in these estimations, and Corner's Calculation can assist with diminishing handling investment utilization in these gadgets.
8. Advanced Channels and Balancers: Computerized channels and adjusters in applications like sound handling and correspondence frameworks require productive augmentation of coefficients with input tests. Stall's Calculation can be utilized to speed up these increases, prompting quicker and more precise sifting activities.
Basically, Corner's Calculation finds its application any place productive paired duplication is required, particularly in situations where speed, power proficiency, and equipment streamlining are significant elements.
Best Case and Worst Case Occurrence: Best case is when there is a large block of consecutive 1's and 0's in the multipliers, so that there is minimum number of logical operations taking place, as in addition and subtraction. Worst case is when there are pairs of alternate 0's and 1's, either 01 or 10 in the multipliers, so that maximum number of additions and subtractions are required.

Memory Hierarchy

A memory unit is an essential component in any digital computer since it is needed for storing programs and data.

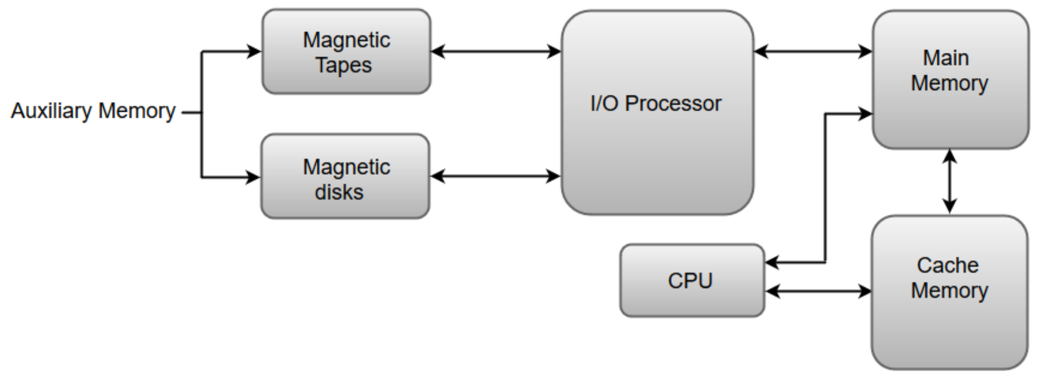
Typically, a memory unit can be classified into two categories:

1. The memory unit that establishes direct communication with the CPU is called **Main Memory**. The main memory is often referred to as RAM (Random Access Memory).
2. The memory units that provide backup storage are called **Auxiliary Memory**. For instance, magnetic disks and magnetic tapes are the most commonly used auxiliary memories.

Apart from the basic classifications of a memory unit, the memory hierarchy consists all of the storage devices available in a computer system ranging from the slow but high-capacity auxiliary memory to relatively faster main memory.

The following image illustrates the components in a typical memory hierarchy.

Memory Hierarchy in a Computer System:



Auxiliary Memory

Auxiliary memory is known as the lowest-cost, highest-capacity and slowest-access storage in a computer system. Auxiliary memory provides storage for programs and data that are kept for long-term storage or when not in immediate use. The most common examples of auxiliary memories are magnetic tapes and magnetic disks.

A magnetic disk is a digital computer memory that uses a magnetization process to write, rewrite and access data. For example, hard drives, zip disks, and floppy disks.

Magnetic tape is a storage medium that allows for data archiving, collection, and backup for different kinds of data.

Main Memory

The main memory in a computer system is often referred to as **Random Access Memory (RAM)**. This memory unit communicates directly with the CPU and with auxiliary memory devices through an I/O processor.

The programs that are not currently required in the main memory are transferred into auxiliary memory to provide space for currently used programs and data.

I/O Processor

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

Cache Memory

The data or contents of the main memory that are used frequently by CPU are stored in the cache memory so that the processor can easily access that data in a shorter time. Whenever the CPU requires accessing memory, it first checks the required data into the cache memory. If the data is found in the cache memory, it is read from the fast memory. Otherwise, the CPU moves onto the main memory for the required data.

We will discuss each component of the memory hierarchy in more detail later in this chapter.

Main Memory

The main memory acts as the central storage unit in a computer system. It is a relatively large and fast memory which is used to store programs and data during the run time operations.

The primary technology used for the main memory is based on semiconductor integrated circuits. The integrated circuits for the main memory are classified into two major units.

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

RAM integrated circuit chips

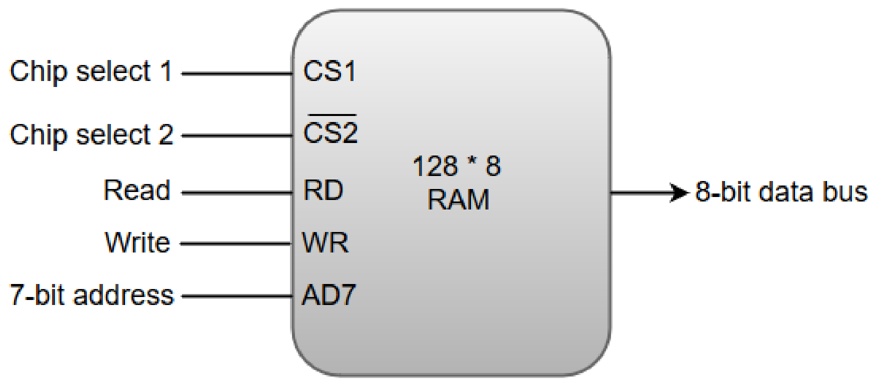
The RAM integrated circuit chips are further classified into two possible operating modes, **static** and **dynamic**.

The primary compositions of a static RAM are flip-flops that store the binary information. The nature of the stored information is volatile, i.e. it remains valid as long as power is applied to the system. The static RAM is easy to use and takes less time performing read and write operations as compared to dynamic RAM.

The dynamic RAM exhibits the binary information in the form of electric charges that are applied to capacitors. The capacitors are integrated inside the chip by MOS transistors. The dynamic RAM consumes less power and provides large storage capacity in a single memory chip.

RAM chips are available in a variety of sizes and are used as per the system requirement. The following block diagram demonstrates the chip interconnection in a 128 * 8 RAM chip.

Typical RAM chip:



- A 128 * 8 RAM chip has a memory capacity of 128 words of eight bits (one byte) per word. This requires a 7-bit address and an 8-bit bidirectional data bus.
- The 8-bit bidirectional data bus allows the transfer of data either from memory to CPU during a **read** operation or from CPU to memory during a **write** operation.
- The **read** and **write** inputs specify the memory operation, and the two chip select (CS) control inputs are for enabling the chip only when the microprocessor selects it.
- The bidirectional data bus is constructed using **three-state buffers**.
- The output generated by three-state buffers can be placed in one of the three possible states which include a signal equivalent to logic 1, a signal equal to logic 0, or a high-impedance state.

ROM integrated circuit

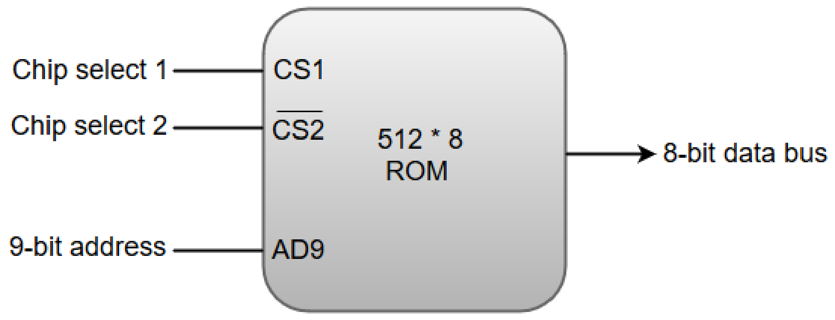
The primary component of the main memory is RAM integrated circuit chips, but a portion of memory may be constructed with ROM chips.

A ROM memory is used for keeping programs and data that are permanently resident in the computer.

Apart from the permanent storage of data, the ROM portion of main memory is needed for storing an initial program called a **bootstrap loader**. The primary function of the **bootstrap loader** program is to start the computer software operating when power is turned on.

ROM chips are also available in a variety of sizes and are also used as per the system requirement. The following block diagram demonstrates the chip interconnection in a 512 * 8 ROM chip.

Typical ROM chip:



- A ROM chip has a similar organization as a RAM chip. However, a ROM can only perform read operation; the data bus can only operate in an output mode.
- The 9-bit address lines in the ROM chip specify any one of the 512 bytes stored in it.
- The value for chip select 1 and chip select 2 must be 1 and 0 for the unit to operate. Otherwise, the data bus is said to be in a high-impedance state.

Auxiliary Memory

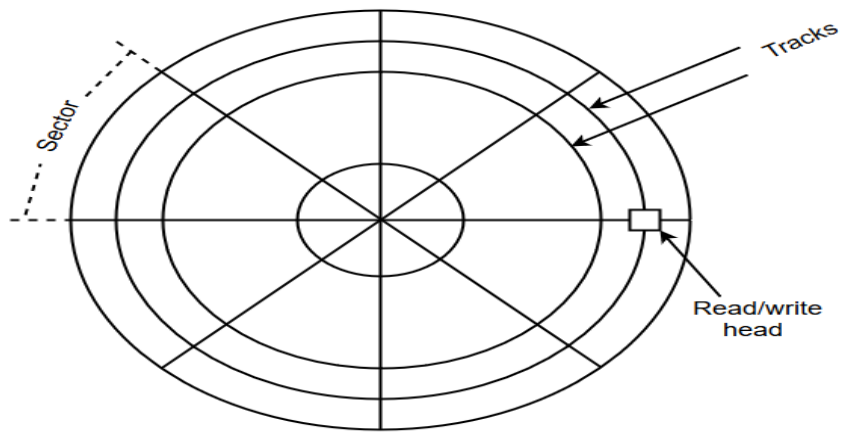
An Auxiliary memory is known as the lowest-cost, highest-capacity and slowest-access storage in a computer system. It is where programs and data are kept for long-term storage or when not in immediate use. The most common examples of auxiliary memories are magnetic tapes and magnetic disks.

Magnetic Disks

A magnetic disk is a type of memory constructed using a circular plate of metal or plastic coated with magnetized materials. Usually, both sides of the disks are used to carry out read/write operations. However, several disks may be stacked on one spindle with read/write head available on each surface.

The following image shows the structural representation for a magnetic disk.

Magnetic disks



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

Magnetic Tape

Magnetic tape is a storage medium that allows data archiving, collection, and backup for different kinds of data. The magnetic tape is constructed using a plastic strip coated with a magnetic recording medium.

The bits are recorded as magnetic spots on the tape along several tracks. Usually, seven or nine bits are recorded simultaneously to form a character together with a parity bit.

Magnetic tape units can be halted, started to move forward or in reverse, or can be rewound. However, they cannot be started or stopped fast enough between individual characters. For this reason, information is recorded in blocks referred to as records.

Associative Memory

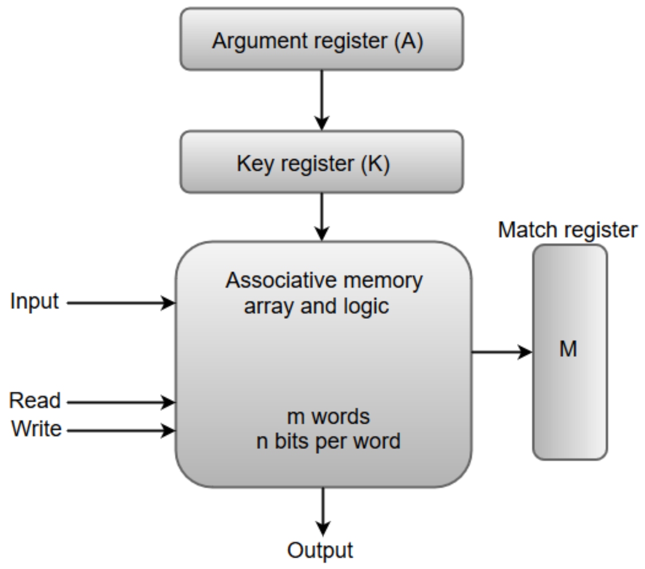
An associative memory can be considered as a memory unit whose stored data can be identified for access by the content of the data itself rather than by an address or memory location.

Associative memory is often referred to as **Content Addressable Memory (CAM)**.

When a write operation is performed on associative memory, no address or memory location is given to the word. The memory itself is capable of finding an empty unused location to store the word.

On the other hand, when the word is to be read from an associative memory, the content of the word, or part of the word, is specified. The words which match the specified content are located by the memory and are marked for reading.

The following diagram shows the block representation of an Associative memory.



From the block diagram, we can say that an associative memory consists of a memory array and logic for 'm' words with 'n' bits per word.

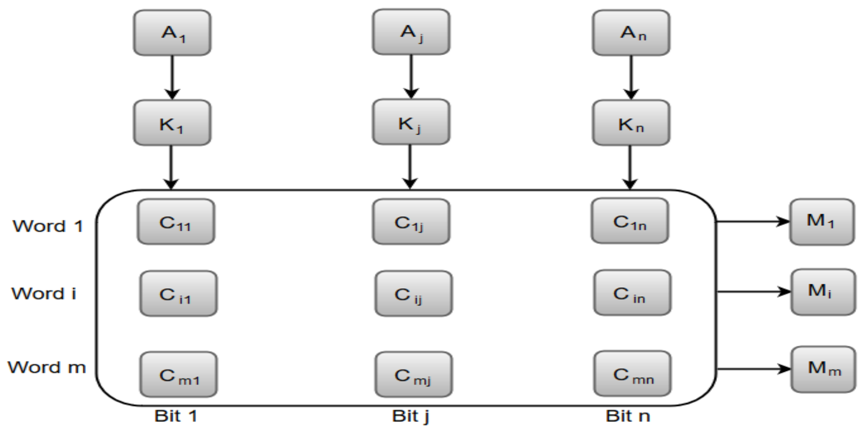
The functional registers like the argument register **A** and key register **K** each have **n** bits, one for each bit of a word. The match register **M** consists of **m** bits, one for each memory word.

The words which are kept in the memory are compared in parallel with the content of the argument register.

The key register (K) provides a mask for choosing a particular field or key in the argument word. If the key register contains a binary value of all 1's, then the entire argument is compared with each memory word. Otherwise, only those bits in the argument that have 1's in their corresponding position of the key register are compared. Thus, the key provides a mask for identifying a piece of information which specifies how the reference to memory is made.

The following diagram can represent the relation between the memory array and the external registers in an associative memory.

Associative memory of m word, n cells per word:



The cells present inside the memory array are marked by the letter C with two subscripts. The first subscript gives the word number and the second specifies the bit position in the word. For instance, the cell C_{ij} is the cell for bit **j** in word **i**.

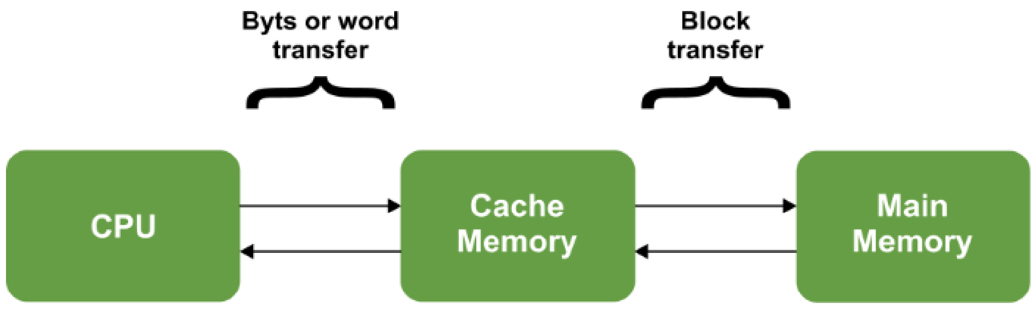
A bit A_j in the argument register is compared with all the bits in column **j** of the array provided that $K_j = 1$. This process is done for all columns **j** = 1, 2, 3......, n.

If a match occurs between all the unmasked bits of the argument and the bits in word **i**, the corresponding bit M_i in the match register is set to 1. If one or more unmasked bits of the argument and the word do not match, M_i is cleared to 0.

Cache Memory

The data or contents of the main memory that are used frequently by CPU are stored in the cache memory so that the processor can easily access that data in a shorter time. Whenever the CPU needs to access memory, it first checks the cache memory. If the data is not found in cache memory, then the CPU moves into the main memory.

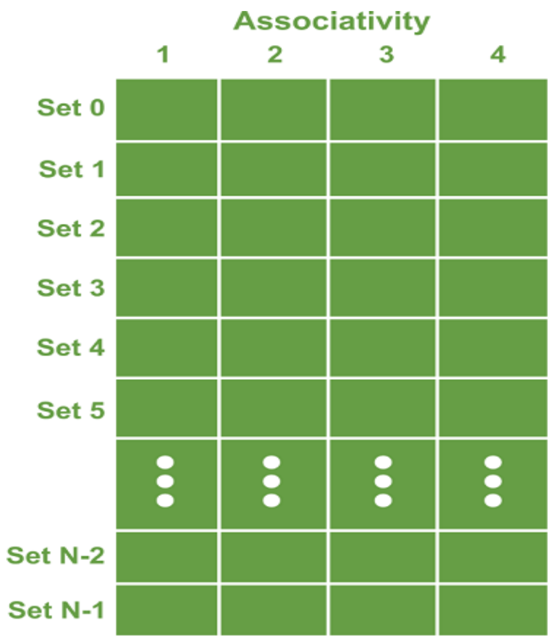
Cache memory is placed between the CPU and the main memory. The block diagram for a cache memory can be represented as:



Cache Memory

The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components.

Cache memory is organised as distinct set of blocks where each set contains a small fixed number of blocks.



Logical organisation of a 4-way set associate cache

As shown in the above sets are represented by the rows. The example contains N sets and each set contains four blocks. Whenever an access is made to cache, the cache controller does not search the entire cache in order to look for a match. Rather, the controller maps the address to a particular set of the cache and therefore searches only the set for a match.

If a required block is not found in that set, the block is not present in the cache and cache controller does not search it further. This kind of cache organisation is called set associative because the cache is divided into distinct sets of blocks. As each set contains four blocks the cache is said to be four way set associative.

The basic operation of a cache memory is as follows:

- 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.
- A block of words one just accessed is then transferred from main memory to cache memory. The block size may vary from one word (the one just accessed) to about 16 words adjacent to the one just accessed.
- 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 the 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.

Levels of memory:

Level 1

It is a type of memory in which data is stored and accepted that are immediately stored in CPU. Most commonly used register is accumulator, Program counter, address register etc.

Level 2

It is the fastest memory which has faster access time where data is temporarily stored for faster access.

Level 3

It is memory on which computer works currently. It is small in size and once power is off data no longer stays in this memory.

Level 4

It is external memory which is not as fast as main memory but data stays permanently in this memory.

Cache Mapping:

There are three different types of mapping used for the purpose of cache memory which are as follows:

- Direct mapping,
- Associative mapping
- Set-Associative mapping

Direct Mapping -
In direct mapping, the cache consists of normal high-speed random-access memory. Each location in the cache holds the data, at a specific address in the cache. This address is given by the lower significant bits of the main memory address. This enables the block to be selected directly from the lower significant bit of the memory address. The remaining higher significant bits of the address are stored in the cache with the data to complete the identification of the cached data.

Virtual Memory in computer architecture

In this chapter of **COA tutorial**, we will learn about:

- *virtual memory in computer architecture*
- *block diagram of virtual memory*
- *advantages of virtual memory in computer architecture*
- *disadvantages of virtual memory in computer architecture*
- *how virtual memory works*?

Virtual memory in computer architecture is used to improve the overall potential and flexibility of advanced computer systems.

What is Virtual Memory?

*Virtual memory in computer organization architecture is a technique and not actually a memory in physical form present in computer system. This is the reason it is known as **virtual memory**.*
Virtual memory in COA is simply a technique used to provide an illusion of presence of large main memory to the programmer, when in actual it's not present physically.
The size of virtual memory is equivalent to the size of secondary memory. Each virtual address or logical address referenced by the CPU is mapped to a physical address in main memory.

A hardware device called ***Memory Management Unit*** (MMU) performs this mapping during run time. To perform this activity MMU actually takes help of a memory map table, which is maintained by the operating system.

What is logical address space and physical address space?

The set of all logical address generated by CPU or program is known as ***logical address space***. It is also called as ***virtual address space***.

The set of all physical addresses corresponding to the above logical addresses is known as ***physical address space***.

The complete procedure is something like this. When a program is required to be executed then the CPU or program would generate a addresses called logical addresses. And executing program occupies corresponding addresses in physical memory called as physical address.
The mapping method in virtual memory uses special register known as relocation register or base register. The content of the base register is added to every logical address generated by user program at the beginning of execution.

Advantages of Virtual memory

Listed below are major *advantages of using virtual memory* techniques:

- Virtual memory technique helps in efficient utilization of main memory. As larger size programs are divided into blocks and partially each block is loaded into main memory as per need. This makes simultaneous execution of multiple program possible.
- Virtual memory helps in efficient CPU utilization
- Virtual memory helps to improve overall throughput.
- Virtual memory usage helps to isolate the processes running on system thus enhancing its security. Its because when every individual process runs in its own virtual space then they won't interfare each other working procedure and modify data of each other.
- Virtual memory use techniques like paging and segmentation to ensure better physical memory utilization.

Lets now look at the disadvantages of virtual memory.

Disadvantages of Virtual Memory

Listed below are major disadvantages of virtual memory:

- The implementation of virtual memory technique is complex in nature. Handling page tables, page faults and translational look aside buffers are complex concepts.
- Virtual memory may face issue known as thrashing. In this issue the system spends more time in swapping the pages in and out of the memory than in actual process or task. This can lead to overall performance degradation of the system.

How virtual memory works?

The virtual memory in computer architecture is implemented such that even though the actual physical memory space is less but it appears to be more while program execution.
A virtual memory can be **configured** using any of the following technique given below:

1. **Paging technique**
2. **Segmentation technique**

Memory Organisation in Computer Architecture

Last Updated : 13 Feb, 2020

-

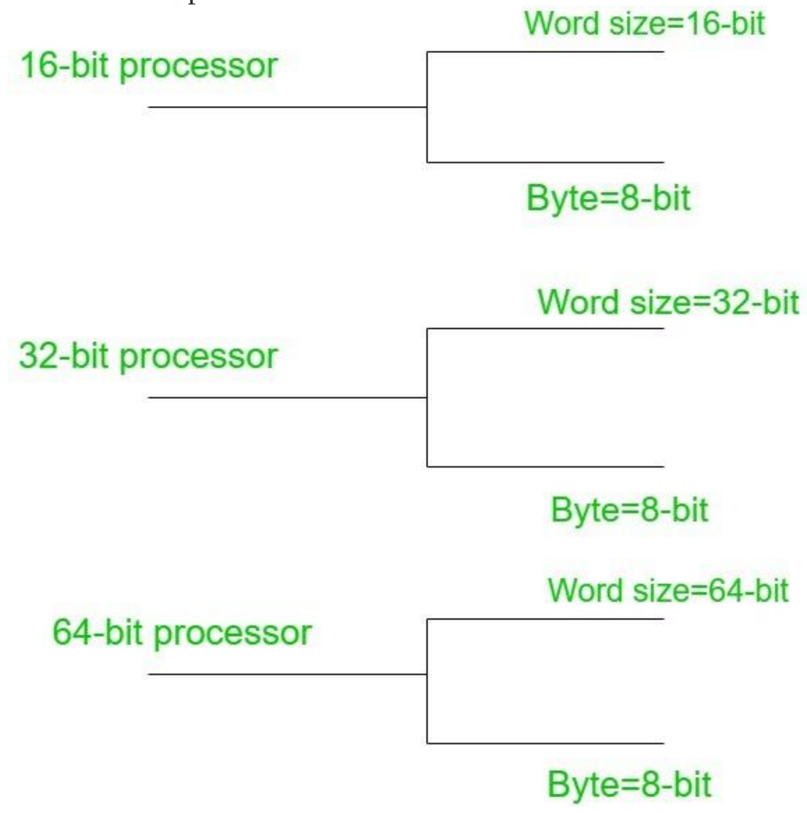
The memory is organized in the form of a cell, each cell is able to be identified with a unique number called address. Each cell is able to recognize control signals such as “read” and “write”, generated by CPU when it wants to read or write address. Whenever CPU executes the program there is a need to transfer the instruction from the memory to CPU because the program is available in memory. To access the instruction CPU generates the memory request.

Memory Request:
Memory request contains the address along with the control signals. For Example, When inserting data into the stack, each block consumes memory (RAM) and the number of memory cells can be determined by the capacity of a memory chip.
Example: Find the total number of cells in 64k*8 memory chip.

```
Size of each cell = 8

Number of bytes in 64k = (2^6)*(2^10)



Therefore,

the total number of cells = 2^16 cells
```

With the number of cells, the number of address lines required to enable one cell can be determined.

Word Size:
It is the maximum number of bits that a CPU can process at a time and it depends upon the processor. Word size is a fixed size piece of data handled as a unit by the instruction set or the hardware of a processor.



Word size varies as per the processor architectures because of generation and the present technology, it could be low as 4-bits or high as 64-bits depending on what a particular processor can handle. Word size is used for a number of concepts like Addresses, Registers, Fixed-point numbers, Floating-point numbers.