Assignment No.: 7

Aim: Vedic Mathematics method to find square of 2-digit number is used in a distributed programming. Use shared memory and distributed (multi-CPU) programming to complete the task.

Objective: To study of Vedic mathematics to find square of 2-digit number is used in a distributed programming.

Theory:

What is Vedic mathematics?

Vedic Mathematics is the name given to the ancient system of Indian Mathematics which was rediscovered from the Vedas between 1911 and 1918 by Sri BharatiKrsnaTirthaji (1884-1960). According to his research all of mathematics is based on sixteen Sutras, or word-formulae. For example, 'Vertically and crosswise` is one of these Sutras. These formulae describe the way the mind naturally works and are therefore a great help in directing the student to the appropriate method of solution.

Perhaps the most striking feature of the Vedic system is its coherence. Instead of a hotchpotch of unrelated techniques the whole system is beautifully interrelated and unified: the general multiplication method, for example, is easily reversed to allow one-line divisions and the simple squaring method can be reversed to give one-line square roots. And these are all easily understood. This unifying quality is very satisfying, it makes mathematics easy and enjoyable and encourages innovation.

In the Vedic system 'difficult' problems or huge sums can often be solved immediately by the Vedic method. These striking and beautiful methods are just a part of a complete system of mathematics which is far more systematic than the modern 'system'. Vedic Mathematics manifests the coherent and unified structure of mathematics and the methods are complementary, direct and easy.

The simplicity of Vedic Mathematics means that calculations can be carried out mentally (though the methods can also be written down). There are many advantages in using a flexible, mental system. Pupils can invent their own methods; they are not limited to the one 'correct' method. This leads to more creative, interested and intelligent pupils.

Interest in the Vedic system is growing in education where mathematics teachers are looking for something better and finding the Vedic system is the answer. Research is being carried out in many areas including the effects of learning Vedic Math's on children; developing new, powerful but easy applications of the Vedic Sutras in geometry, calculus, computing etc. But the real beauty and effectiveness of Vedic Mathematics cannot be fully appreciated without actually practicing the system. One can then see that it is perhaps the most refined and efficient mathematical system possible.

Sixteen (16) Sutras of Vedic Mathematics

Sr. No.	Sutras	Meaning
1.	EkadhikinaPurvena	By one more than the previous one
2.	NikhilamNavatashcaramamDashatah	All from 9 and the last from 10
3.	Urdhva-Tiryagbyham	Vertically and crosswise
4.	ParaavartyaYojayet	Transpose and adjust
5.	ShunyamSaamyasamuccaye	When the sum is the same that sum is zero.
6.	(Anurupye) Shunyamanyat	If one is in ratio, the other is zero
7.	Sankalana-vyavakalanabhyam	By addition and by subtraction
8.	Puranapuranabyham	By the completion or non-completion
9.	Chalana-Kalanabyham	Differences and Similarities
10.	Yaavadunam	Whatever the extent of its deficiency
11.	Vyashtisamanstih	Part and Whole
12.	ShesanyankenaCharamena	The remainders by the last digit
13.	Sopaantyadvayamantyam	The ultimate and twice the penultimate
14.	EkanyunenaPurvena	By one less than the previous one
15.	Gunitasamuchyah	The product of the sum is equal to the sum of the product
16.	Gunakasamuchyah	The factors of the sum is equal to the sum of the factors

1. EkadhikenaPurvena

The Sutra (formula) EkadhikenaPūrvena means: “By one more than the previous one”.

- Squares of numbers ending in 5 :

Now we relate the sutra to the ‘squaring of numbers ending in 5’. Consider the example 252

Here the number is 25. We have to find out the square of the number. For the number 25, the last digit is 5 and the 'previous' digit is 2. Hence, 'one more than the previous one', that is, 2+1=3. The Sutra, in this context, gives the procedure to multiply the previous digit 2 by one more than itself, that is, by 3. It becomes the L.H.S (left hand side) of the result, that is, 2 X 3 = 6. The R.H.S (right hand side) of the result is 52, that is 25. Thus 252= 2 X 3 / 25 = 625. In the same way,

352= 3 X (3+1) /25 = 3 X 4/ 25 = 1225;

652= 6 X 7 / 25 = 4225;

1052= 10 X 11/25 = 11025;

1352= 13 X 14/25 = 18225;

Share Memory:

In computer programming, shared memory is a method by which program processes can exchange data more quickly than by reading and writing using the regular operating system services. For example, a client process may have data to pass to a server process that the server process is to modify and return to the client. Ordinarily, this would require the client writing to an output file (using the buffers of the operating system) and the server then reading that file as input from the buffers to its own work space. Using a designated area of shared memory, the data can be made directly accessible to both processes without having to use the system services. To put the data in shared memory, the client gets access to shared memory after checking a semaphore value, writes the data, and then resets the semaphore to signal to the server (which periodically checks shared memory for possible input) that data is waiting. In turn, the server process writes data back to the shared memory area, using the semaphore to indicate that data is ready to be read.

Distributed Memory:

Distributed memory refers to a multiple-processor computer system in which each processor has its own private memory. Computational tasks can only operate on local data, and if remote

data is required, the computational task must communicate with one or more remote processors. In contrast, a shared memory multi processor offers a single memory space used by all processors. Processors do not have to be aware where data resides, except that there may be performance penalties, and that race conditions are to be avoided.

Mathematical Model:

Let this complete system represented mathematically.

M={I, O, P, Sc, Fc} where

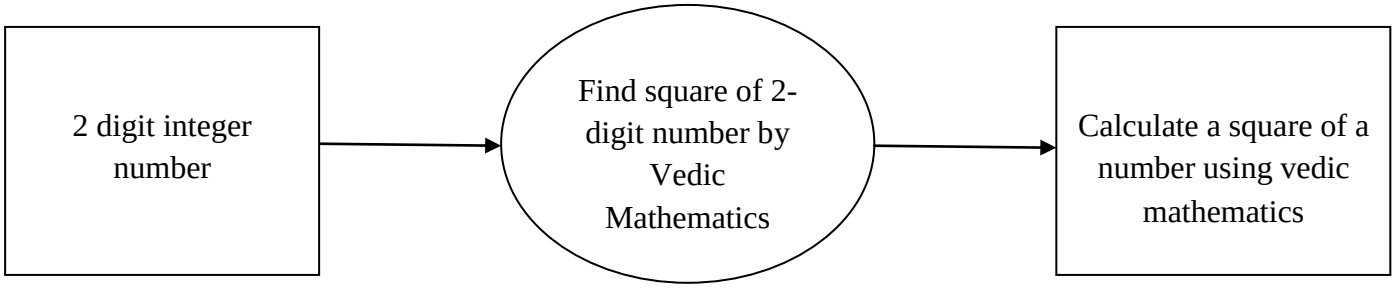
I = Input

O = Output

P = Process

Sc = Success cases.

Fc = Failure cases.



Input **Process** **Output**

Input: Client providing 2 digit integer number

Processing: Find square of 2- digit number by distributed programming is used in vedic maths.

Output: Server calculate a square of a number using vedic

Success case:

- Output produce correct result.
- Work properly shared memory.
- Function work properly

Failure case:

- Incorrect result.
- Syntax error

Platform/ languages: Ubuntu 12.04

Conclusion: Thus we successfully calculated square of 2-digit number using vedic mathematics method.

FAQs:

1. How to attach and detach the shared memory portion in distributed shared memory?

Ans: shmat() and shmdt() are used to attach and detach shared memory segments. They are prototypes as follows:

*void *shmat(int shmid, const void *shmaddr, int shmflg);*

*int shmdt(const void *shmaddr);*

shmat() returns a pointer, shmaddr, to the head of the shared segment associated with a valid shmid. shmdt() detaches the shared memory segment located at the address indicated by shmaddr.

2. Shared memory is logical memory or physical memory?

Ans: Shared memory is physical memory that is assigned to the shared memory pool and shared among multiple logical partitions. The shared memory pool is a defined collection of physical memory blocks that are managed as a single memory pool by the hypervisor. Logical partitions that you configure to use shared memory (hereafter referred to as shared memory partitions) share the memory in the pool with other shared memory partitions.

(i) When shared memory partitions are not actively using their memory pages, the hypervisor allocates those unused memory pages to shared memory partitions that currently need them. When the sum of the physical memory currently used by the shared memory partitions is less than or equal to the amount of memory in the shared memory pool, the memory configuration is logically overcommitted. In a logically overcommitted memory configuration, the shared memory pool has

enough physical memory to contain the memory used by all shared memory partitions at one point in time. The hypervisor does not need to store any data in auxiliary storage.

(ii) When a shared memory partition requires more memory than the hypervisor can provide to it by allocating unused portions of the shared memory pool, the hypervisor stores some of the memory that belongs to a shared memory partition in the shared memory pool and stores the remainder of the memory that belongs to the shared memory partition in auxiliary storage. When the sum of the physical memory currently used by the shared memory partitions is greater than the amount of memory in the shared memory pool, the memory configuration is physically overcommitted. In a physically overcommitted memory configuration, the shared memory pool does not have enough physical memory to contain the memory used by all the shared memory partitions at one point in time. The hypervisor stores the difference in auxiliary storage. When the operating system attempts to access the data, the hypervisor might need to retrieve the data from auxiliary storage before the operating system can access it.

3. What are the different commands we can use to control the shared memory portion?

Ans: Different commands that we can use to control the shared memory portion are:

i. shmget():

Creates the shared memory block.

ii. shmat():

Maps an existing shared memory block into a process's address space.

iii. shmdt():

Removes (unmaps) a shared memory block from the process's address space.

iv. shmctl():

It is a general-purpose function (in the style of ioctl()) used to perform all other commands to the shared memory block.