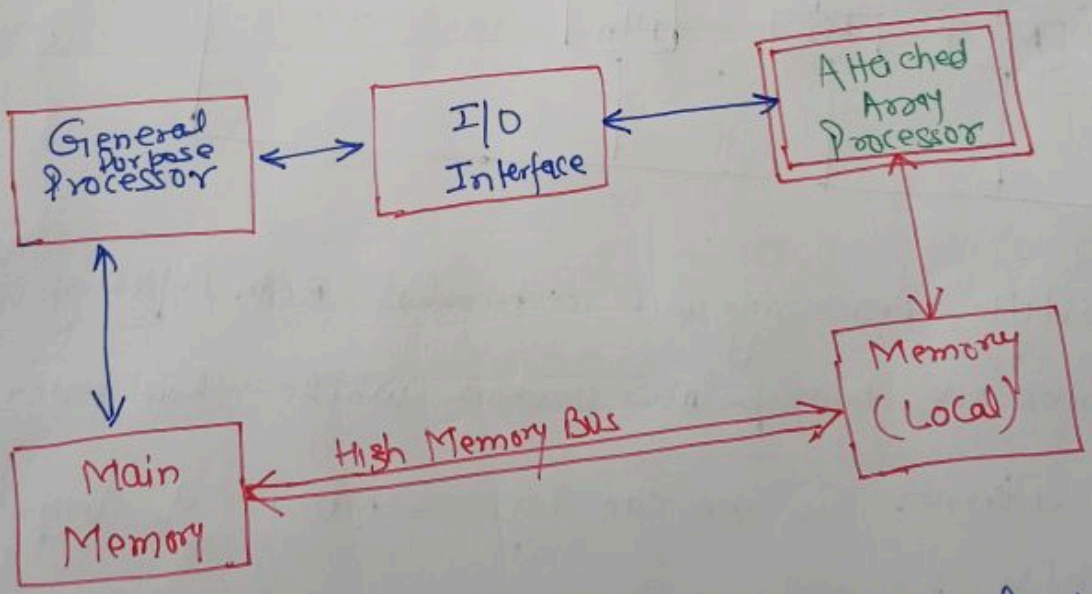
Array Processor when we create a number of processor to form a array, we create Array Processor environment. An array Processor is essentially a Synchronous array of parallel Processors, where each processor consists of multiple processing elements (PEs, which are basically ALUs) operated under the supervision of one control unit.

There are two type of Array processors. ① Attached Array Processor ② SIMD Array Processor

① Attached Array Processor:

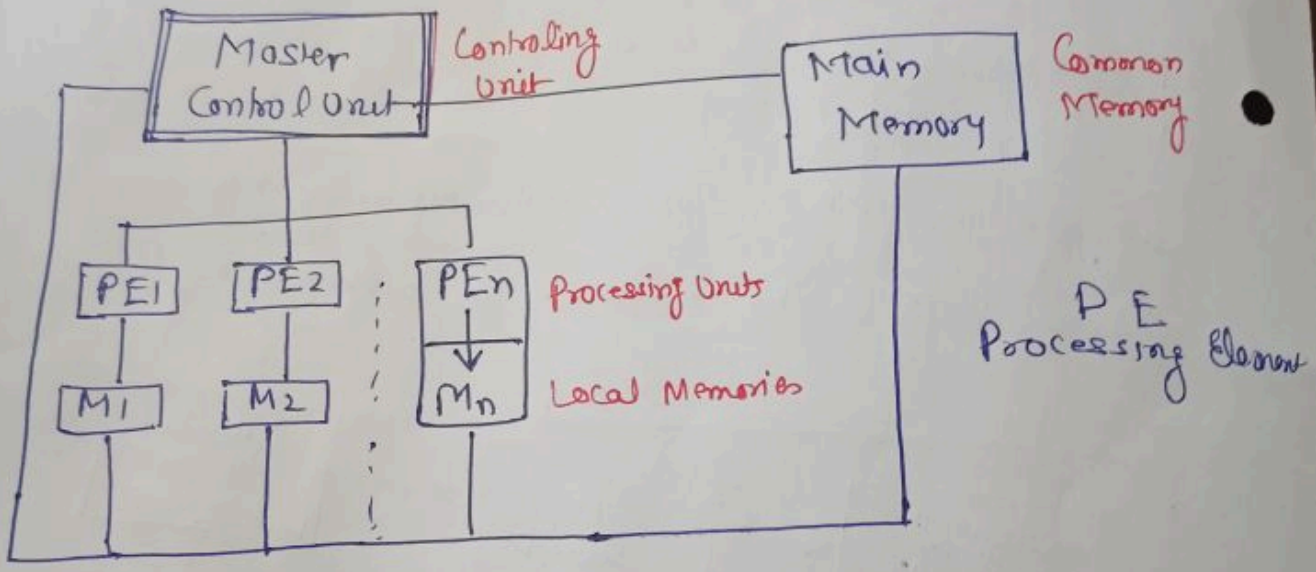
We use Array Processor when we data on numerical Computation tasks, we use Auxiliary processor as attached form to enhance the performance.



For better performance, Improvement of the host (General purpose Computer) in numerical Computational work/task Auxilliary Processor is attached, we call it Attached Array processor.

Attached array processor has two interfaces:
① Input output interface to a common processor
② Interface with a local Memory.

② SIMD Array Processor — Single Instruction multiple Data



Multiple Processing unit in parallel Both types of array processors, manipulate vectors while their internal organisation is different. So we can say SIMD is a computer with multiple processing units operating in parallel.

PE are synchronized to perform same operation under the control of a common control unit.

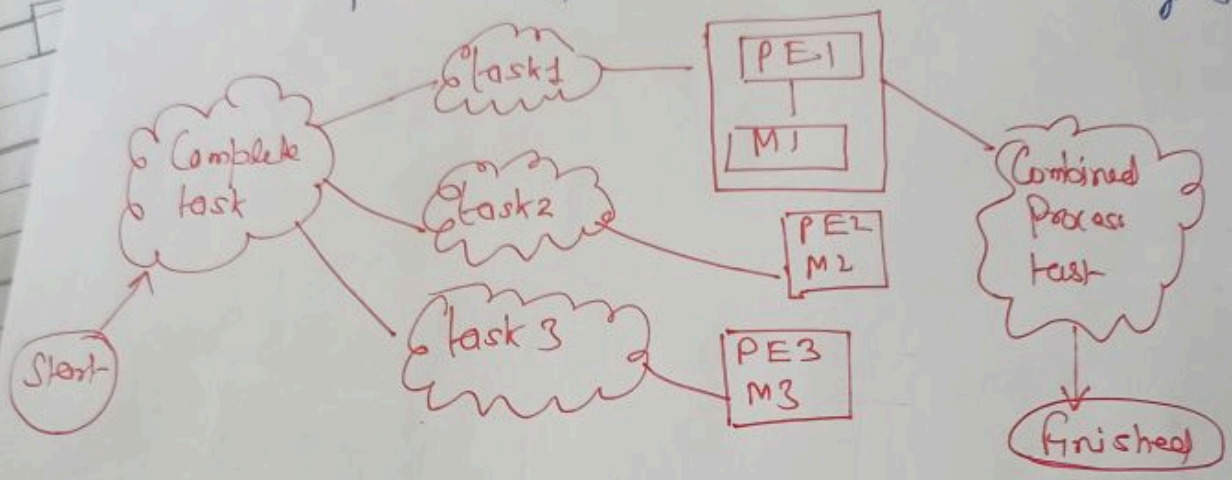
Each PE includes:-
① ALU
② Floating ~~unit~~ Point Arithmetic Unit
③ Working registers.

Master Unit decode the instruction and determine how the Instruction will be executed.

Main Memory is used for storage of the program while each PE uses operands stored in its local Memory

Flynn's Taxonomy

Parallel Computing is a computing where the jobs are broken with discrete parts that can be executed concurrently. Each part is further broken down to a series of instruction. Instructions from each part execute simultaneously on

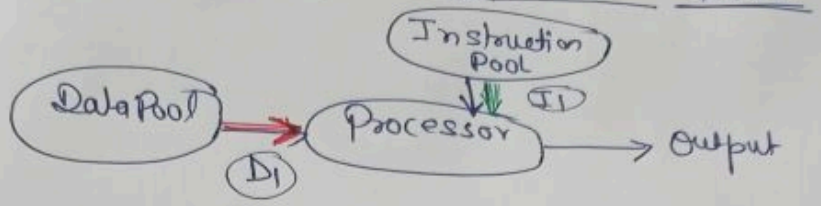


different CPU. Parallel Computers Systems are more difficult to program, distribution of works, Synchronisation and clubbing all parallel work to Consolidate into Combined work.

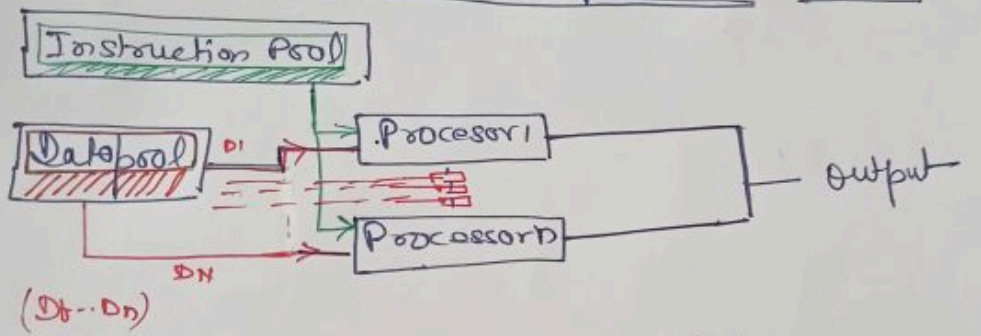
Flynn's classification

Data Streams \ Instruction Streams	One	Many
One	(Von Neuman) SISD (Single CPU System)	MISD we can say Pipelined Computers
Many	SIMD Vector Processing Parallel Computers	MIMD - Multi Complete Computers - Multi processors

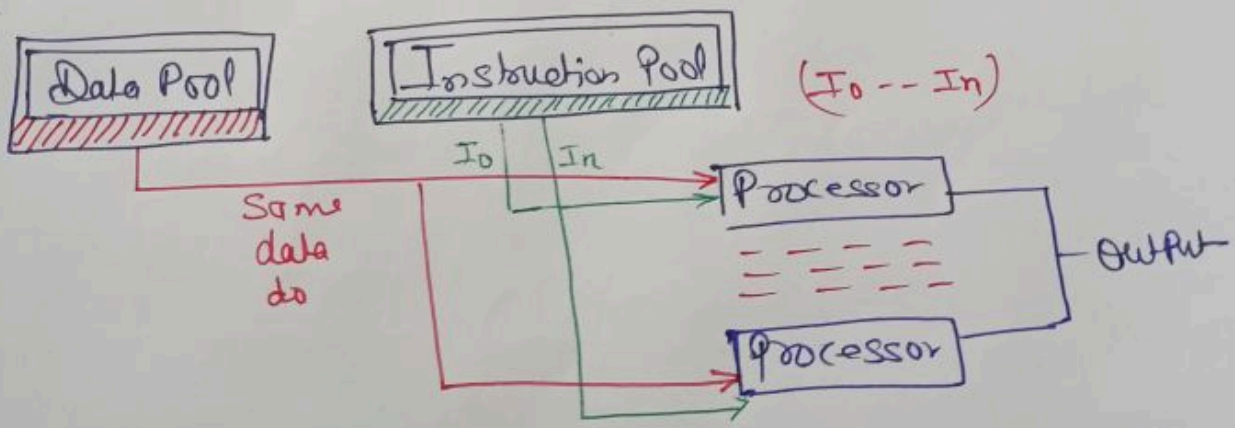
① Single Instruction Single Data SISD



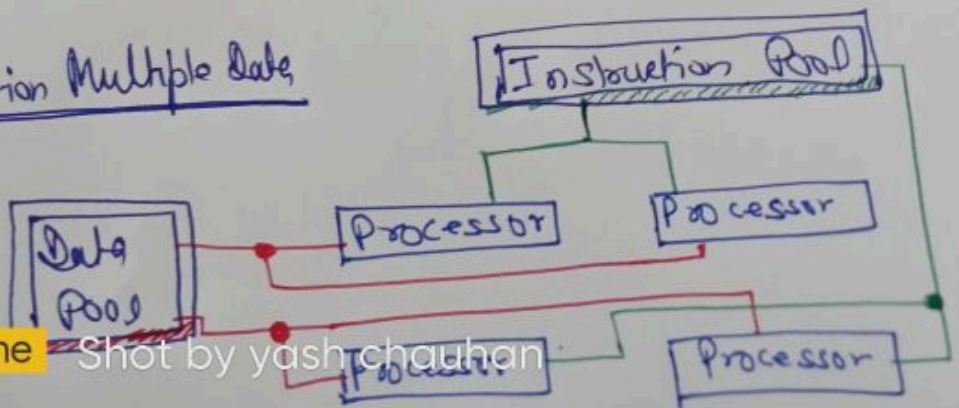
② Single Instruction Multiple data SIMD

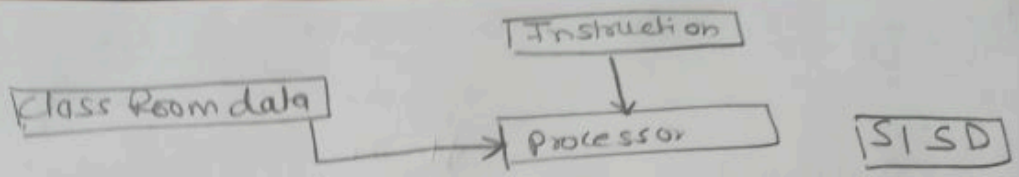


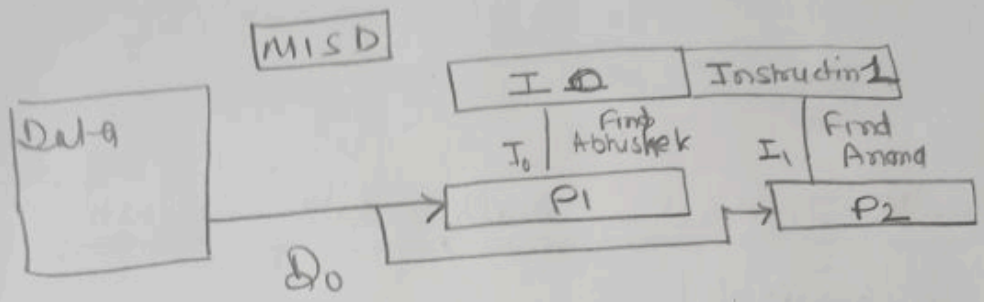
③ Multiple - Instruction Single-data MISD

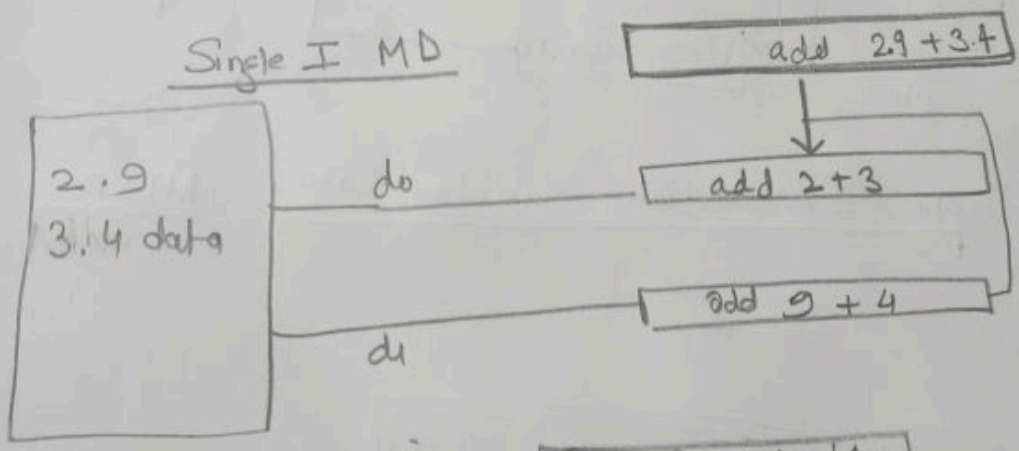


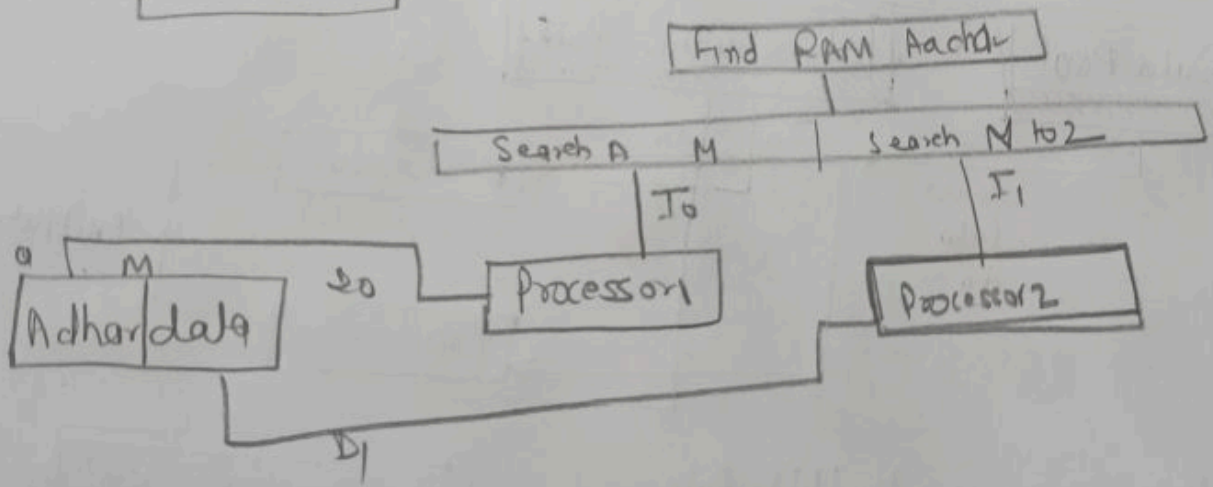
④ Multiple Instruction Multiple Data









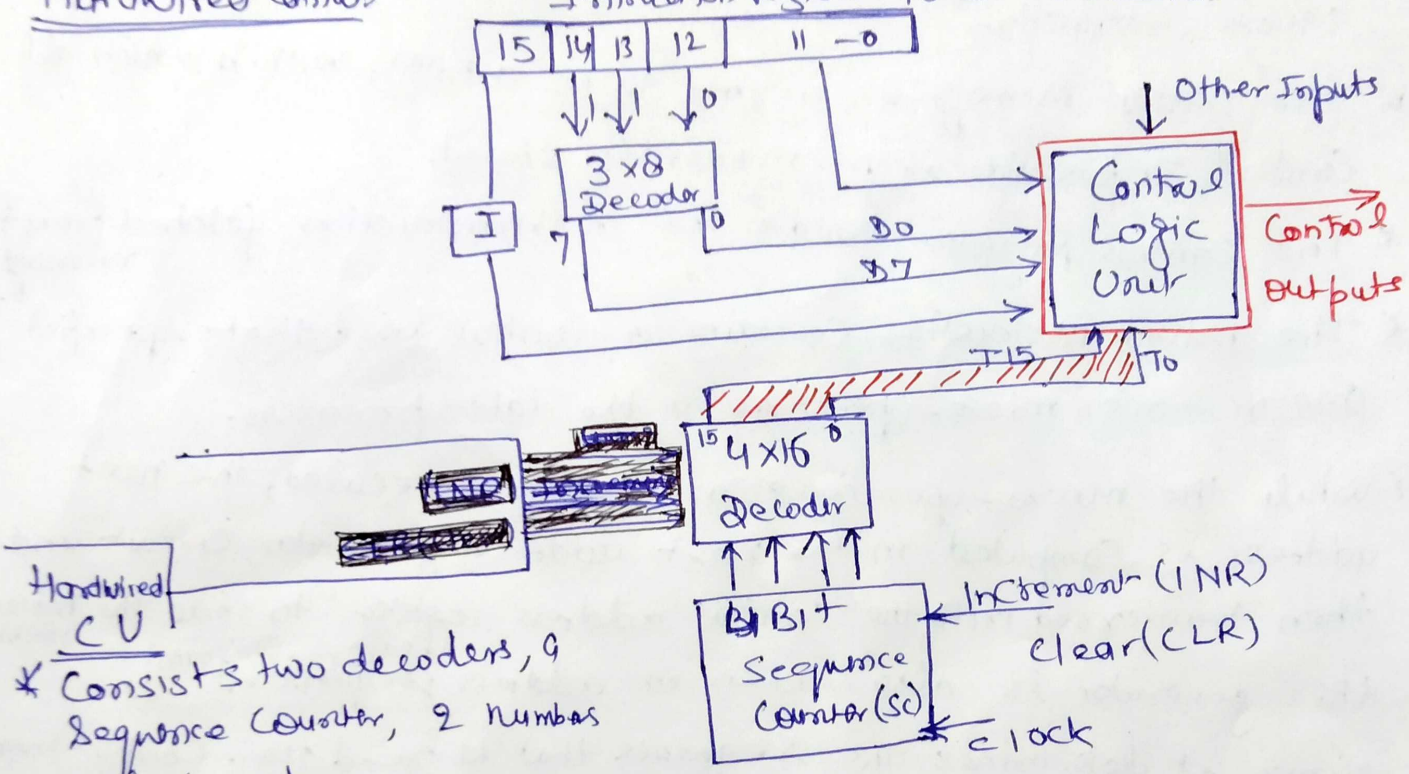


(CU) Control Unit is the part of the Computer's Central processing Unit (CPU) which directs the operation of the processor. CU responsibility is to tell memory, arithmetic/logic unit and input and output devices, how to respond to the Instructions that is available in main memory and bring them to the processor Instruction Register (IR), based on the IR Contents the Control Unit generate a Control Signal that Supervises the execution of these Instructions.

Design of Control Unit

Classified into two major Categories
① Hardwired Control ② Microprogrammed Control

Hardwired Control

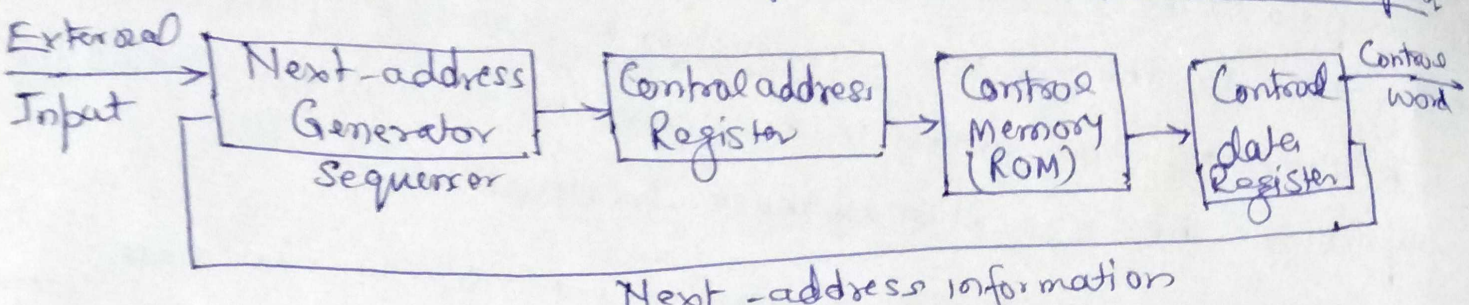


* Consists two decoders, 4 Sequence Counter, 2 numbers of logic gates.
* Instruction fetched from memory Unit is placed in the Instruction register.
* Components of IR includes, I bit, the operation code 4 bits 8 to 11
* 12 to 14 bits decoded with 3x8 decoder, the decoded output given to Con Control Logic Unit D0 to D7.
* Operation Code from Bits 0 through 11 to produce Control signal output
* But 15 is transferred to a flip-flop, designated by the Symbol I. →

Micro-Programmed Code

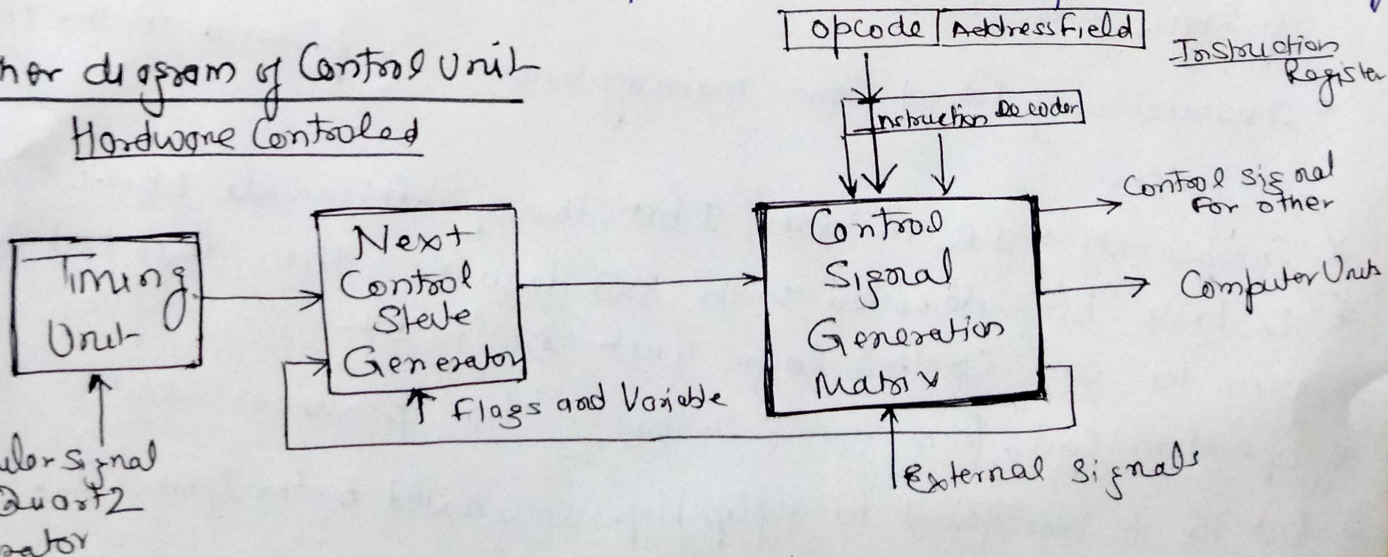
The micro-operation are performed by a program consisting of micro-Instructions.

Microprogrammed Unit of a Basic Computer

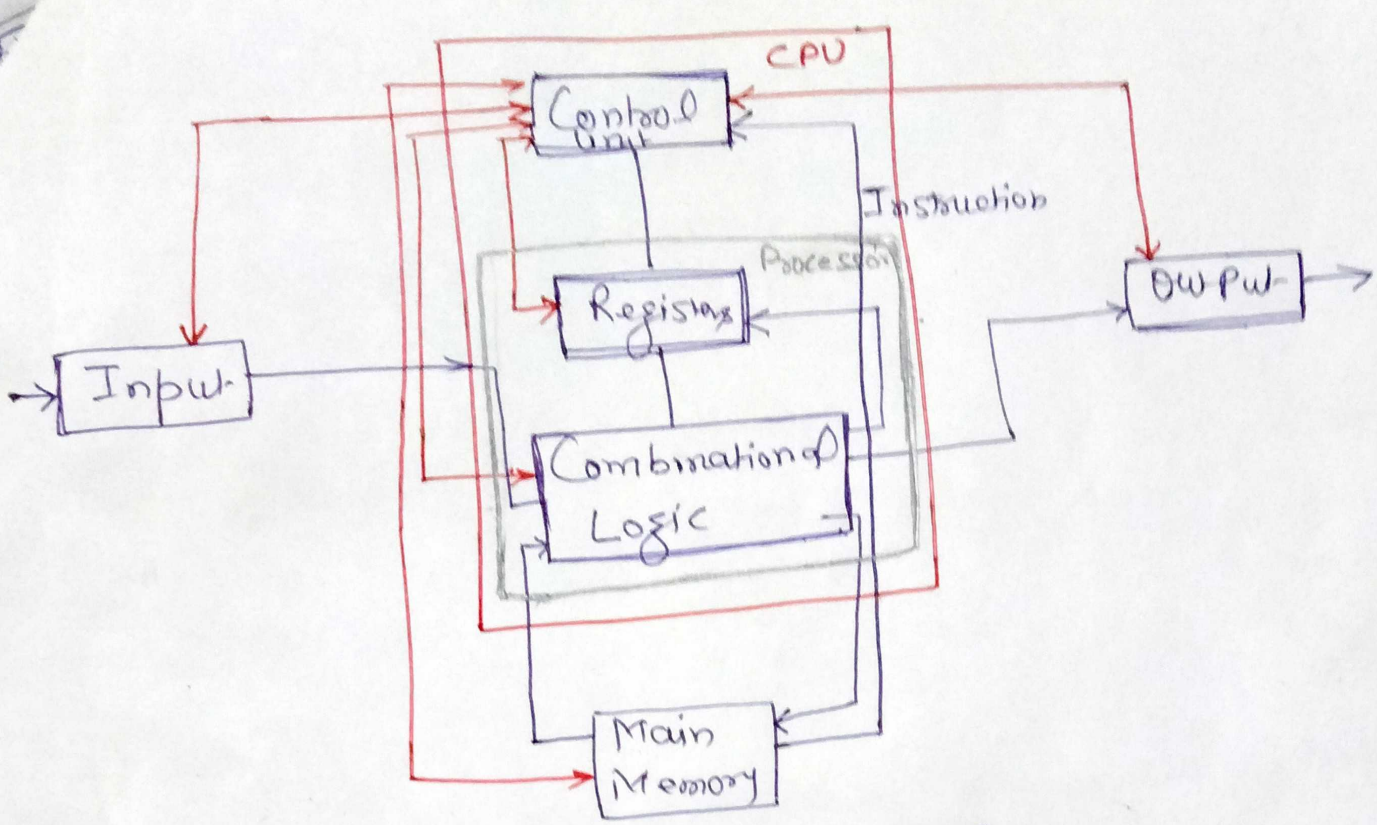


* The Control Memory address register specifies the address of the Micro-Instruction
* The Control Memory is assumed to be a ROM, within which all Control Information is permanently stored.
* The Control Register holds the microinstruction fetched from the memory.
* The micro-instruction contains a control word that specified one or more micro-operations for the data processor.
* While the micro-operations are being executed, the next address is computed in the next address generator circuit and then transferred into the control address register to read the micro instruction. this generator is also known as address Micro-Program Sequencer, Since it determines the sequence that is read from control memory.

another diagram of Control Unit Hardware Controlled

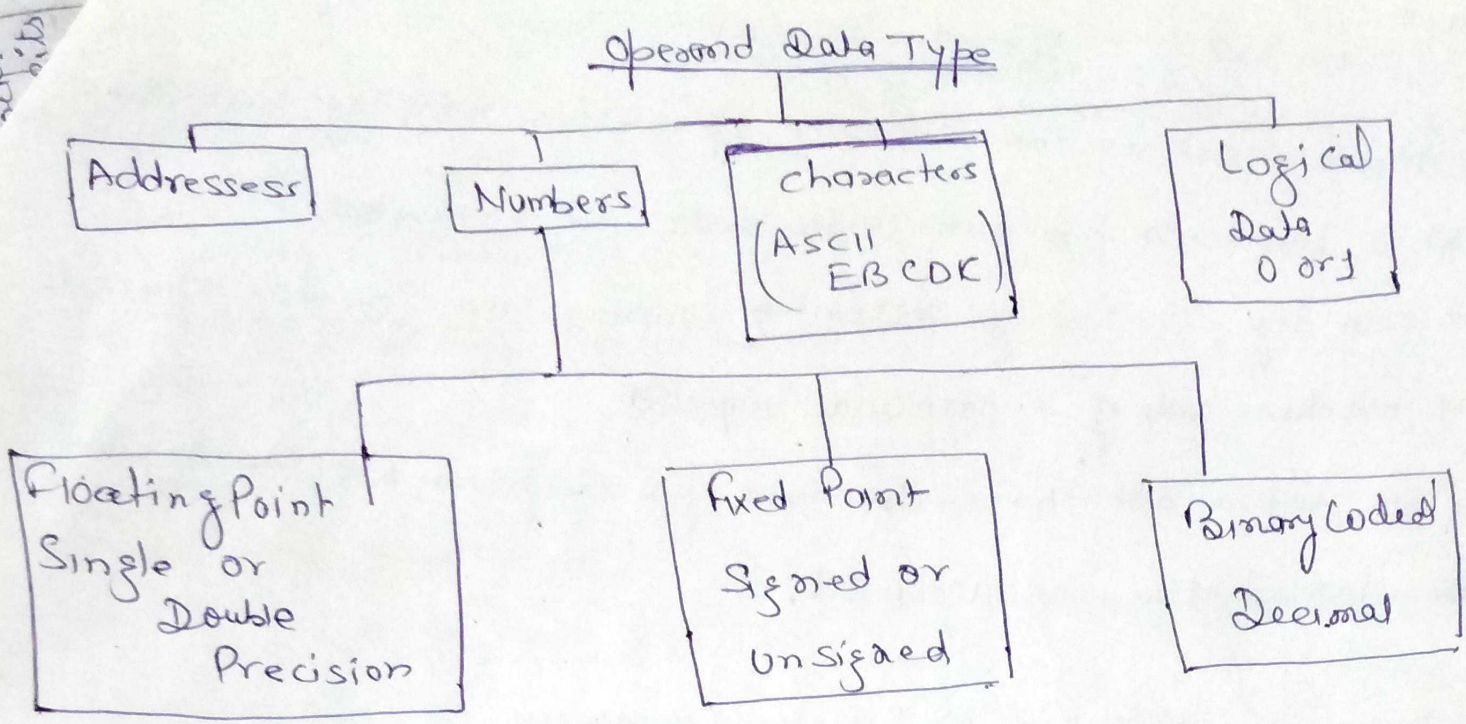


CPU



Block diagram of CPU

Computer Architecture is a set of rules and methods that describe the functionality, organization and Implementation of Computer Systems.



Addresses: operands residing in memory are specified by their memory address and operands residing in registers are specified by a register address. Addresses provided in the instruction are operand references.

Numbers: All machine language include numeric data types. Numeric data usually use one of the three representations:

- Floating-Point numbers - Single precision (1 Sign bit, 8 Exponent bits, 23 mantissa bits)

  Double Precision (1 Sign bits, 52 mantissa bits)

- Fixed point number (signed or unsigned).

Instruction Set

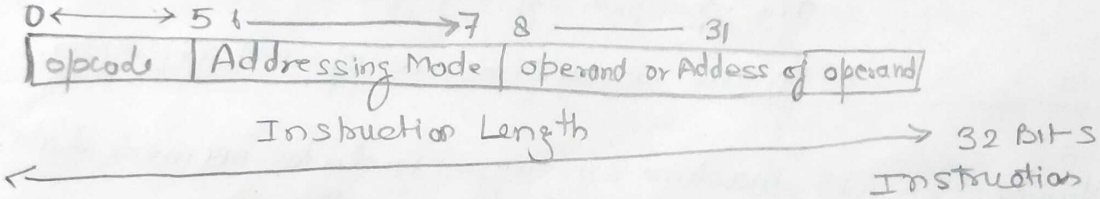
Instruction set is the collection of machine language instructions that a particular processor understands and executes.

We can say, a set of assembly language mnemonics represents the machine code of a particular computer.

If we define all the instructions of a computer, we can say we have defined the instruction set.

Format of Instruction

1. Length of Instruction
2. Type
3. Length of operation codes and et position
4. Number and length of operand addresses etc.

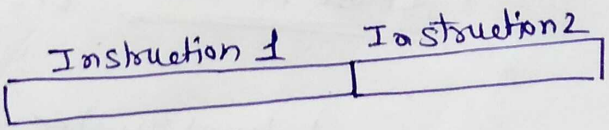


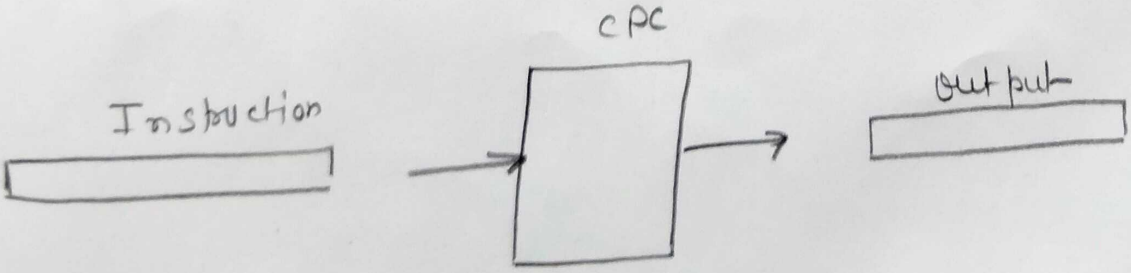
Who make Instruction

1. Compiler
2. Interpretor
3. Translators.

Program Code

```
C = a + b ;
C = 2 * C ;
```





What are the Essential elements of Instruction?

The purpose of Instruction Set (IS) is to communicate CPU - what to do.

There should be in IS,

- what operation to perform?
- on what operands?

OPCode An operation code

Operation Code field known as opcode, Specifies the operation to be performed.

different Architecture allow different operation

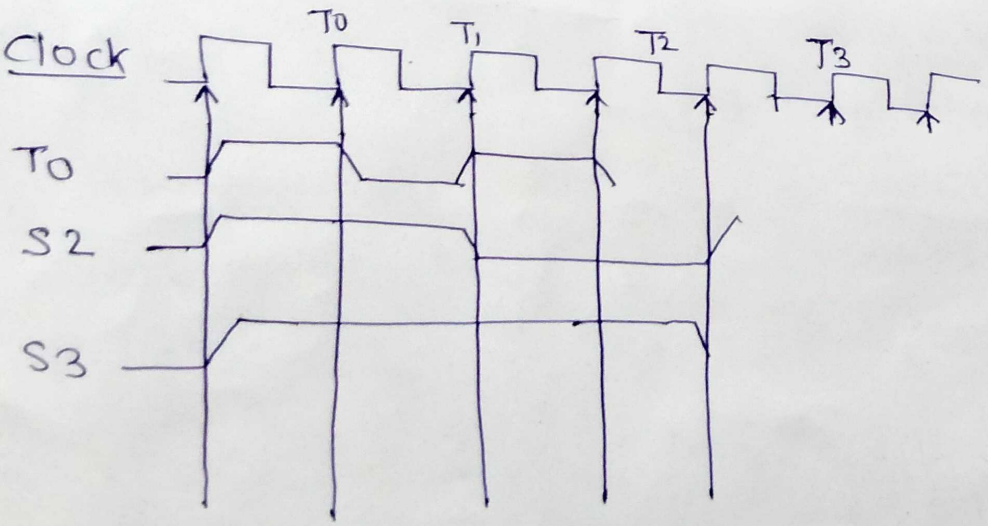
The most common are

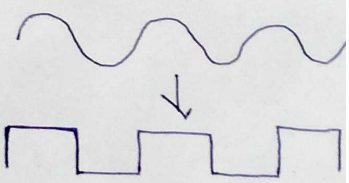
* Add
* Substract
* Shift left
* Shift Right

ALU Arithmatic logic Unit performs the arithmatic and logical operations:

- Addition
- Substraction
- Logical AND
- Logical or
- Logical Exclusive OR
- Complement (Logical NOT)
- Increment (add 1)
- Decrement (Substract 1)
- Left Shift, Rotate left, Rotate right
- Clear

Electrical pulse generated in the processor or in external devices in order to synchronize computer operations. The main timing signal comes from computer's clock (master clock) which provides a frequency that can be divided into many slower cycles. Other timing signals may come from a timesharing or real-time-clock.





Managing highspeed with lower speed clock

Difference between Computer Architecture and Computer Organization:

S.NO	Computer Architecture	Computer Organization
1.	Architecture describes what the computer does.	The Organization describes how it does it.
2.	Computer Architecture deals with the functional behavior of computer systems.	Computer Organization deals with a structural relationship.
3.	In the above figure, it's clear that it deals with high-level design issues.	In the above figure, it's also clear that it deals with low-level design issues.
4.	Architecture indicates its hardware.	Where Organization indicates its performance.
5.	For designing a computer, its architecture is fixed first.	For designing a computer, an organization is decided after its architecture.
6.	Computer Architecture is also called instruction set architecture.	Computer Organization is frequently called microarchitecture.
7.	Computer Architecture comprises logical functions such as instruction sets, registers, data types, and addressing modes.	Computer Organization consists of physical units like circuit designs, peripherals, and adders.
8.	Architecture coordinates between the hardware and software of the system.	Computer Organization handles the segments of the network in a system.

3.2.1. Shift-and-Add Multiplication

Shift-and-add multiplication is similar to the multiplication performed by paper and pencil. This method adds the multiplicand X to itself Y times, where Y denotes the multiplier. To multiply two numbers by paper and pencil, the algorithm is to take the digits of the multiplier one at a time from right to left, multiplying the multiplicand by a single digit of the multiplier and placing the intermediate product in the appropriate positions to the left of the earlier results.

As an example, consider the multiplication of two unsigned 4-bit numbers, 8 (1000) and 9 (1001).

```
Multiplicand        1000 ×
Multiplier          1001
                    1000
                   0000
                  0000
                 1000
Product         1001000
```

In the case of binary multiplication, since the digits are 0 and 1, each step of the multiplication is simple. If the multiplier digit is 1, a copy of the multiplicand (1 × multiplicand) is placed in the proper positions; if the multiplier digit is 0, a number of 0 digits (0 × multiplicand) are placed in the proper positions.

Consider the multiplication of positive numbers. The first version of the multiplier circuit, which implements the shift-and-add multiplication method for two n-bit numbers, is shown in Figure 3.11.

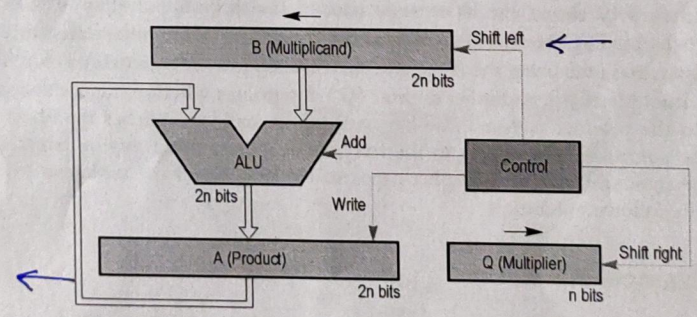
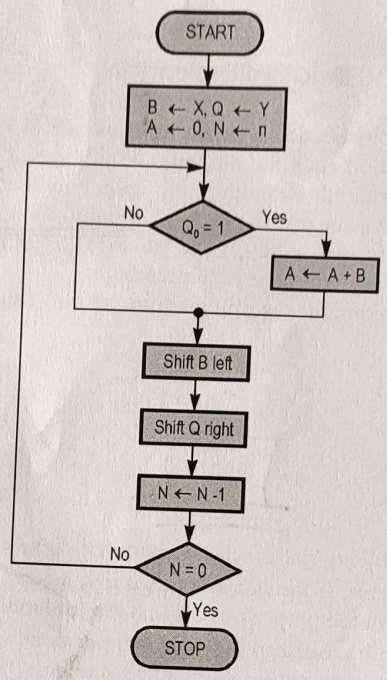


Figure 3.11. First version of the multiplier circuit.

The $2n$-bit product register (A) is initialized to 0. Since the basic algorithm shifts the multiplicand register (B) left one position each step to align the multiplicand with the sum being accumulated in the product register, we use a $2n$-bit multiplicand register with the multiplicand placed in the right half of the register and with 0 in the left half.



Not use it X

Figure 3.12. The first version of the multiplication algorithm.

Figure 3.12 shows the basic steps needed for the multiplication. The algorithm starts by loading the multiplicand into the B register, loading the multiplier into the Q register, and initializing the A register to 0. The counter N is initialized to n. The least significant bit of the multiplier register (Q_0) determines whether the multiplicand is added to the product register. The left shift of the multiplicand has the effect of shifting the intermediate products to the left, just as when multiplying by paper and pencil. The right shift of the multiplier prepares the next bit of the multiplier to examine in the following iteration.

Example 3.1

Using 4-bit numbers, perform the multiplication 9×12 (1001×1100).

Answer

Table 3.2 shows the value of registers for each step of the multiplication algorithm.

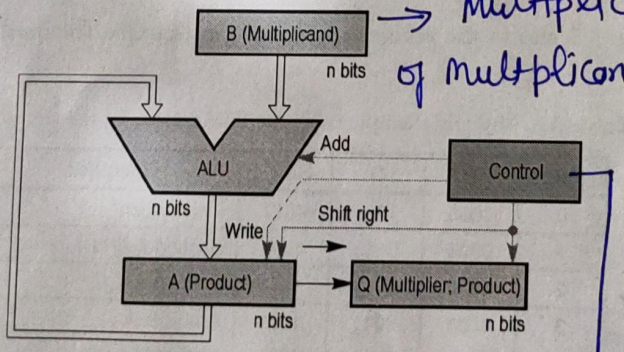
Table 3.2. Multiply example using the first version of the algorithm.

Step	A	Q	B	Operation
0	0000 0000	1100	0000 1001	Initialization
1	0000 0000	1100	0001 0010	Shift left B
	0000 0000	0110	0001 0010	Shift right Q
2	0000 0000	0110	0010 0100	Shift left B
	0000 0000	0011	0010 0100	Shift right Q
3	0010 0100	0011	0010 0100	Add B to A
	0010 0100	0011	0100 1000	Shift left B
	0010 0100	0001	0100 1000	Shift right Q
4	0110 1100	0001	0100 1000	Add B to A
	0110 1100	0001	1001 0000	Shift left B
	0110 1100	0000	1001 0000	Shift right Q

Not to use of

The original algorithm shifts the multiplicand left with zeros inserted in the new positions, so the least significant bits of the product cannot change after they are formed. Instead of shifting the multiplicand left, we can shift the product to the right. Therefore the multiplicand is fixed relative to the product, and since we are adding only n bits, the adder needs to be only n bits wide. Only the left half of the $2n$-bit product register is changed during the addition.

Another observation is that the product register has an empty space with the size equal to that of the multiplier. As the empty space in the product register disappears, so do the bits of the multiplier. In consequence, the final version of the multiplier circuit combines the product (A register) with the multiplier (Q register). The A register is only n bits wide, and the product is formed in the A and Q registers. Figure 3.13 shows the new version of the circuit.

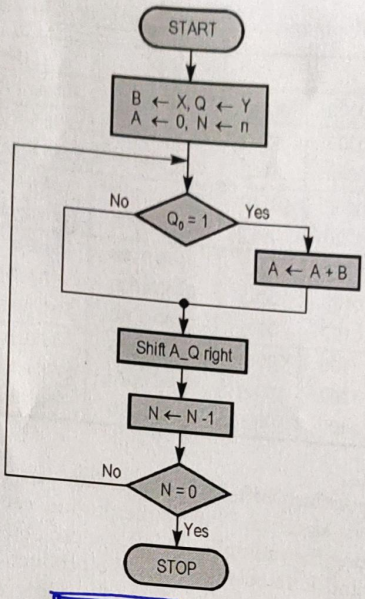


Multiplicand will Remain Same of multiplicand perform every Time Instruction will be applied in loop every time.

Figure 3.13. Final version of the multiplier circuit.

The final version of the multiplication algorithm is shown in Figure 3.14.

dependency of code Control will decide whether ALU Add or Write without add



Multiplicand B

Multiplier Q

Figure 3.14. The final version of the multiplication algorithm.

Example 3.2

Perform the multiplication 9×12 (1001×1100) using the final version of the multiplication algorithm.

Answer

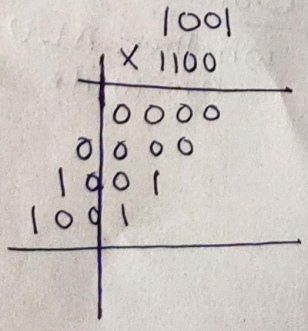
Table 3.3 shows the revised multiplication example for the final version of the algorithm.

Table 3.3. Multiply example using the final version of the algorithm.

Step	A	Q	B	Operation
0	0000	1100	1001	Initialization
1	0000	0110	1001	Shift right A_Q
2	0000	0011	1001	Shift right A_Q
3	1001	0011	1001	Add B to A
	0100	1001	1001	Shift right A_Q
4	1101	1001	1001	Add B to A
	0110	1100	1001	Shift right A_Q

when 1 come then A ← A+B

B remain same



$$\begin{vmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{vmatrix} \times \begin{vmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{vmatrix} = \begin{vmatrix} C_{11} & C_{12} & C_{13} \\ C_{21} & C_{22} & C_{23} \\ C_{31} & C_{32} & C_{33} \end{vmatrix}$$

$C_{11} = (A_{11} \times B_{11}) + (A_{12} \times B_{21}) + (A_{13} \times B_{31})$

.
.
.

In general

$C_{ij} = \sum (A_{ik} \times B_{kj})$ (for k=1 to 3)