Unit 9 : Fundamentals of Parallel Processing

Lesson 1 : Types of Parallel Processing

1.1. Learning Objectives

On completion of this lesson you will be able to :

- classify different types of parallel processing techniques.

Traditionally, the computer has been viewed as a sequential machine. Most computer programming languages require the programmer to specify algorithms as sequences of instructions. CPUs execute programs by executing machine instructions in a sequence and one at a time. Each instruction is executed in a sequence of operation (fetch instruction, fetch operands, perform operation, store results).

CPUs execute programs by executing machine instructions in a sequence and one at a time.

This view of the computer has never been entirely true. At the micro-operation level, multiple control signals are generated at the same time. Instruction pipelining, at least to the extent of overlapping fetch and execute operations, has been around for a long time. Both of these are examples of performing functions in parallel.

When a computer application requires a very large amount of computation that must be completed in a reasonable amount of time, it becomes necessary to use machines with correspondingly large computing capacity. Such machines are called supercomputers. Typical applications that require supercomputers include weather forecasting; simulation of large, complex, physical systems; and computer-aided design (CAD) using high-resolution graphics.

As a general quantitative measure, a supercomputer should have the capability to execute at least 100 million instructions per second.

A supercomputer execute at least 100 million instruction per second.

In the development of powerful computers, two basic approaches can be followed. The first possibility is to build the system around a high-performance single processor, and to use the fastest circuit technology available. Architectural features such as multiple functional units for arithmetic operations, pipelining, large caches, and separate buses for instructions and data can be used to achieve very high throughput. As the design is based on a single processor, the system is relatively easy to use because conventional software techniques can be applied.

The second approach is to configure a system that contains a large number of conventional processors.

The second approach is to configure a system that contains a large number of conventional processors. The individual processors do not have to be complex, high-performance units. They can be standard microprocessors. The system derives its high performance form the fact that many computations can proceed in parallel.

1.2. Classification of Parallel Structures

Flynn classification of parallel processors.

A general classification of forms of parallel processing has been proposed by Flynn. Machines are categorized according to the way that data and instructions are related. According to the Flynn classification there are four basic types :

- Single instruction stream - single data stream (SISD)
- Single instruction stream - multiple data steam (SIMD)
- Multiple instruction stream - single data steam (MISD)
- Multiple instruction stream - multiple data steam (MIMD).

1.2.1. Single Instruction Stream - Single Data Stream (SISD)

A single-processor computer system is called a single instruction stream, single data stream (SISD) system.

In this classification, a single-processor computer system is called a single instruction stream, single data stream (SISD) system. A program executed by the processor constitutes the single instruction stream, the sequence of data items that it operates on constitutes the single data stream. This computer design is the basis for the traditional general purpose computer and is called a von Neumann computer. A von Neumann machine fetches an instruction from memory, decodes the instruction, and then executes it. When the action on an instruction is complete, the next instruction follows the same fetch-decode-execute sequence until a stop occurs, as shown in Fig. 9.1.

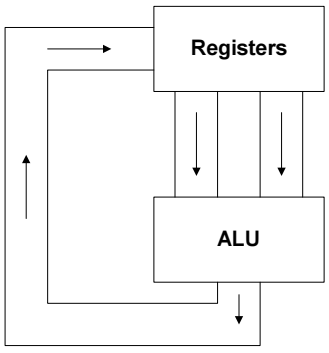


Fig. 9.1 : A simple von Neumann computer.

1.2.2. Single Instruction Stream - Multiple Data Stream (SIMD)

This scheme, in which all processors execute the same program, is called a single Instruction stream.

A single stream of instructions is broadcasted to a number of processors. Each processor operator on its own data. This scheme, in which all processors execute the same program, is called a single Instruction stream, multiple data stream (SIMD) system. This type includes machines supporting array parallelism.

1.2.3. Multiple Instruction Stream: Multiple Data Stream (MIMD)

This scheme involves a number of independent processors are called multiple instruction stream.

This scheme involves a number of independent processors, each executing a different program and accessing its own sequence of data items. Such machines are called multiple instruction stream multiple data stream (MIMD) system. This type covers the multiprocessor systems supporting process parallelism.

1.2.4. Multiple Instruction Stream Single : Data Stream (MISD)

In such a system, a common data structure is manipulated by separate processors and each executes a different program. This form of computation does not arise often in practice. Hence, no system have been built which fit this classification.

As computer technology has evolved, and as the cost of computer hardware has dropped, computer designers have sought more and more opportunities for parallelism, usually to improve performance and, in some cases, to improve reliability.

In the subsequent lessons we shall examines multiprocessing, one of the earliest and still the most common example of parallel organization. Typically, multiprocessing involves the use of multiple CPUs sharing a common memory. Next, we will look at hardware organizational approaches to vector computation. These approaches optimize the ALU for processing vectors or arrays of floating-point numbers. They have been used to implement the class of systems know as supercomputers. Finally, we will look at the more general area referred to as parallel processor organization.

1.3. Exercise

1.3.1. Multiple choice questions

a) A Von Neumann computer uses which one of the following?

i) SISD
ii) SIMD
iii) MISD
iv) MIMD.

1.3.2. Questions for short answer

a) Classify different types of parallel processing.

1.3.3. Analytical question

a) Write down the classification of parallel structures.

Lesson 2 : Pipelined Vector Processors

2.1. Learning Objectives

On completion of this lesson you will be able to :

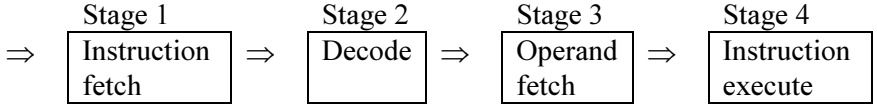
- learn about pipelined vector processors.

Many of the early attempts at exploiting parallelism in computer architecture were based on the use of pipelining. The ATLAS computer, designed and built at the University of Manchester, UK. In the late 1950s, was one of the earliest computers to make use of pipelining.

Pipelining is a well-known technique for using several pieces of hardware concurrently.

Pipelining is a well-known technique for using several pieces of hardware concurrently. Thus the system performance and overall system throughput capability of a system increases by increasing the number of instructions that may be completed per second. The intention behind this technique is to subdivide a computational task into several sub-tasks, each one handled by a separate hardware stage. A series of such hardware stages is called a pipeline. All stages operate simultaneously with independent inputs. A computational task advances from one stage to the next, and comes closer to completion as the end of the pipeline is approached the pipeline accepts new inputs before previously accepted inputs have been completely processed and output from it. When one sub-task result leaves a stage, that stage can accept new results from the previous stage. Once the pipline is full, the output rate will match the input rate. The processing of any instruction can be subdivided into the following four sub-tasks: instruction fetch, instruction decode, operand fetch and instruction execute. Suppose instruction # 1 is fetched. While it is decoded, instruction # 2 is fetched. While instruction # 1 fetches operand, instruction # 2 fetches operand, instruction # 3 is decoded and instruction # 4 is fetched. This process continues until all the instructions are executed. Fig. 9.2 shows the process of pipelining.

A true pipeline implementation establishes a series of four hardware stages, in which each stage operates concurrently with the others on one of these sub-tasks of several instructions.



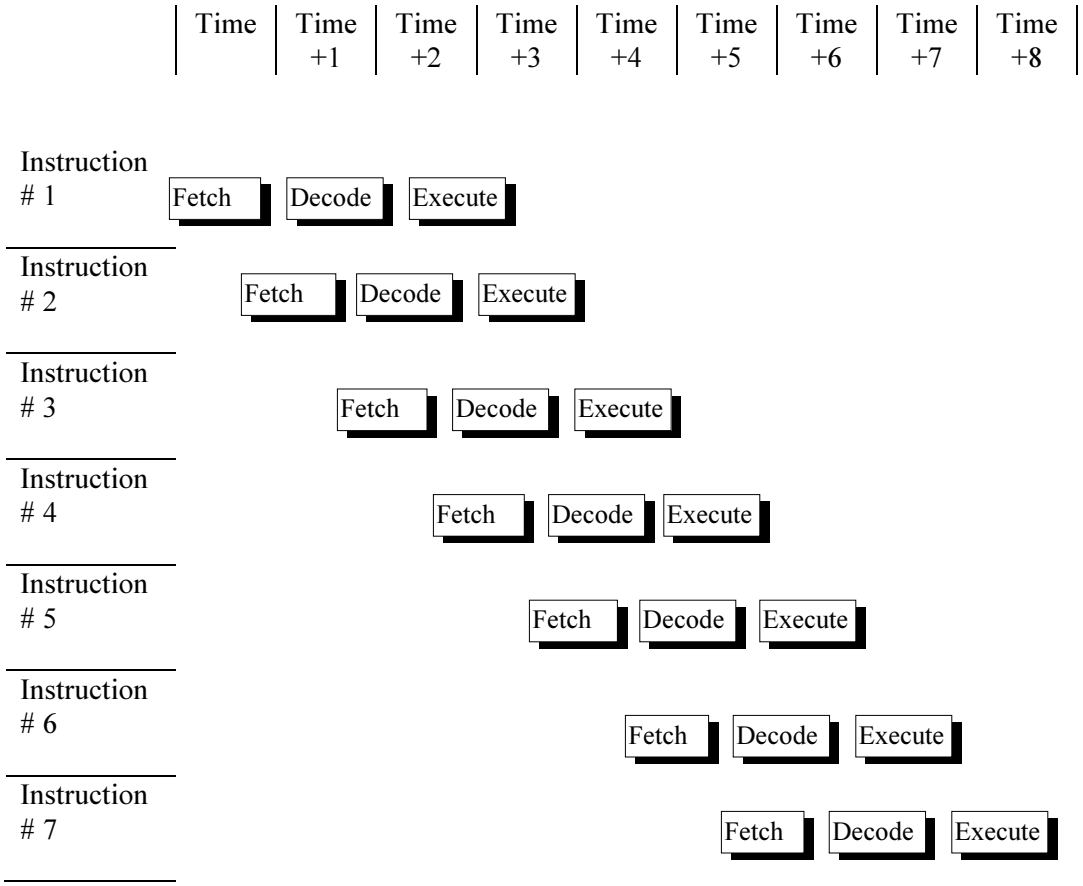


Fig. 9.2 : Process of pipelining.

Process of pipelining.

Vector pipelines are simply the extension of the idea of pipelining to vector instructions with vector operands. They form a predictable, regular, sequence of scalar operations on the components of the vectors. As a simple example, suppose that we wish to form the vector sum $c \leftrightarrow a + b$ and that the addition of two scalars can be divided into three stages, each requiring one clock to execute then.

- on the first clock cycle.
  - a_1 and b_1 are in stage 1 of the addition
- on the second clock cycle
  - a_1 and b_1 are in stage 2 of the addition and
  - a_2 and b_2 are in stage 1
- on the third clock cycle
  - a_1 and b_1 are in stage 3 of the addition
  - a_2 and b_2 are in stage 2 and
  - a_3 and b_3 are in stage 1.

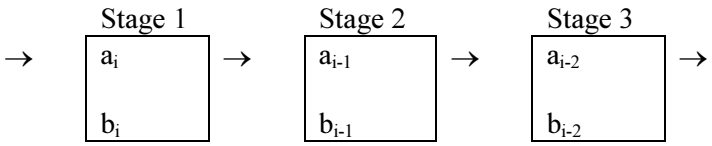


Fig. 9.3 : Status of the pipeline on the i-th clock cycle.

Status of the pipeline.

There after all these stages of the pipeline are kept busy until the final components and enter the pipe. At the end of the i-th clock cycle, illustrated in Fig. 9.3, the result ci-2 ← ai-2 + bi-2 is produced.

A vector pipeline processor is an implementation, in hardware, of the pipelining just described. Vector pipelining forms the basis of many of today's vector super computers, such as the Cray family and the Japanese (Fuji's, Hitachi and NEC) supercomputers. The Cray supercomputers have several processors and a number of pipeline units per processor and hence support both process and pipeline parallelism.

2.2. Exercise

2.2.1. Questions for short answer

a) Which are the subtasks of pipeline operation?

2.2.2. Analytical question

a) Discuss the operation of pipelined vector processor system.

Lesson 3 : Array Processor

3.1. Learning Objectives

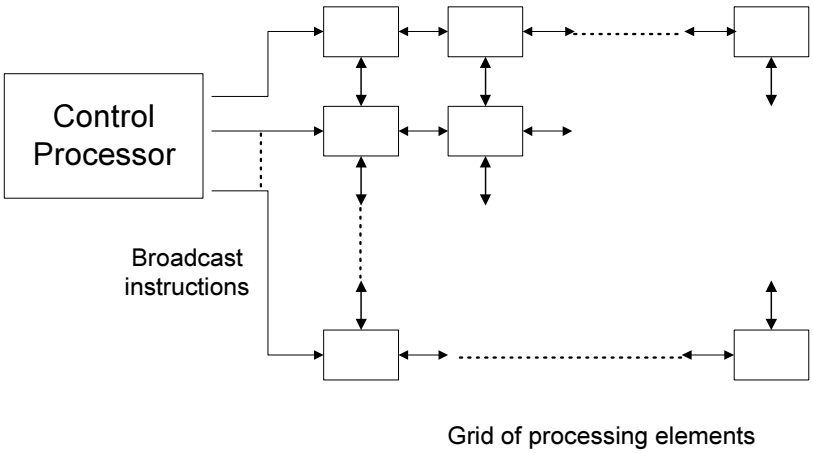
On completion of this lesson you will be able to :

- learn about array processor.

An array processor is of SIMD type processors connected together to form a grid.

One version of a parallel computer is the array processor. An array processor (or data parallel machine) is of SIMD type and consists of a number of identical and relatively elementary processors connected together to form a grid, which is often square. Each active processor (some may be idle) obeys the same instruction, issued by a single control unit but on different local data. As there is a single control unit, there is a single program which operates on all the data at once. Thus there are no difficulties with synchronization of the processors. Array processors are particularly well suited to computations involving matrix manipulations. Examples of array processors are the Active Memory Technology (AMT), Distributed Array processor (DAP). Thinking Machines, connection machine and the MasPar MP-1.

Array processor has its own memory but all the processors share the same control unit. Array processors are usually in a grid formation of rows and columns and work well for matrix applications. A two dimensional grid of processing elements executes an instruction stream broadcast from a central control processor. As each instruction is broadcast, all elements is connected to its four nearest neighbors for purposes of exchanging data. End around connections may be provided on both rows and columns.



Array processor.

Fig. 9.4 : Array processor.

The grid of processing elements can be used to generate solutions to two-dimensional problems.

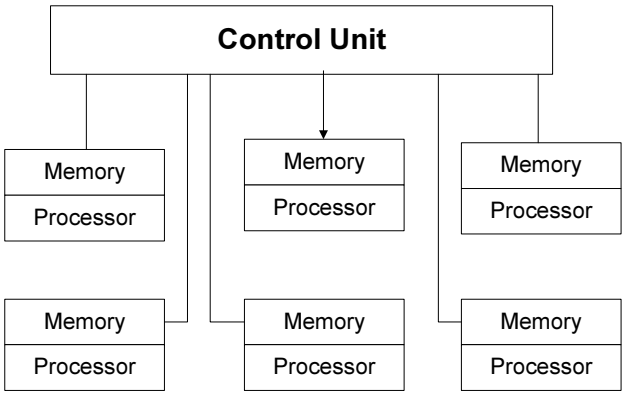


Fig. 9.5 : Example of an array processor.

Fig. 9.5 shows 2×3 array: each unit has local memory and a processor with shared control. Each processing element has a dew registers and local memory to store data. It also has a register, which is called network register. This register facilitate movement of values to and from its neighbors. The control processor can broadcast an instruction to shift the values in the network registers one step up, down, left or right. Each processing element also contains an ALU to execute arithmetic instructions broadcast by the control processor. Using these basic facilities, a sequence of instructions can be broadcast repeatedly to implement the iterative loop. Array processors are highly specialized machines. They are well suited to numerical problems that can be expressed in matrix or vector format. However, they are not very useful in speeding up general computations.

3.2. Exercise

3.2.1. Multiple choice questions

a) An array processor has its

i) own memory
ii) different memory
iii) two memory
iv) all of the above.

b) Array processors one highly specialized

i) computers
ii) machines
iii) diodes
iv) none of the above.

3.2.2. Questions for short answer

a) What is an array processor?

3.3.3. Analytical question

a) Describe the function of an array processor.

Lesson 4 : Multiprocessor Systems

4.1. Learning Objectives

On completion of this lesson you will be able to

- learn about multi-processor systems.

Multiprocessor system involves a number of processors capable of independently executing different routines in parallel.

Multiprocessor system involves a number of processors capable of independently executing different routines in parallel. This type of system supports MIMD architecture. A general MIMD configuration, usually called a multiprocessor is shown in Fig. 9.6.

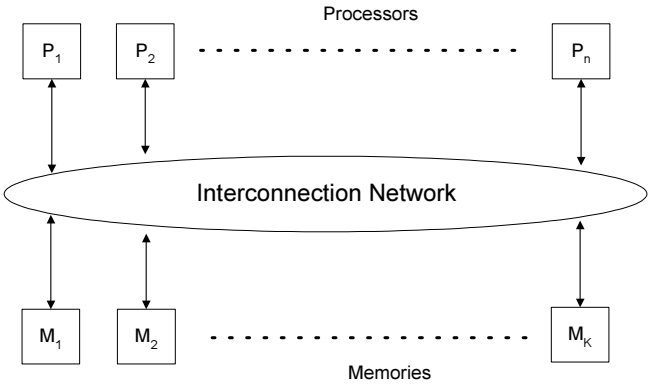


Fig. 9.6 : A general multiprocessor.

An interconnection network permits n processors to access K memories, so that any of the processors can access any of the memories.

An alternative arrangement, which allows a high computation rate to be sustained in all processors, is to attach some local memory directly to each processor.

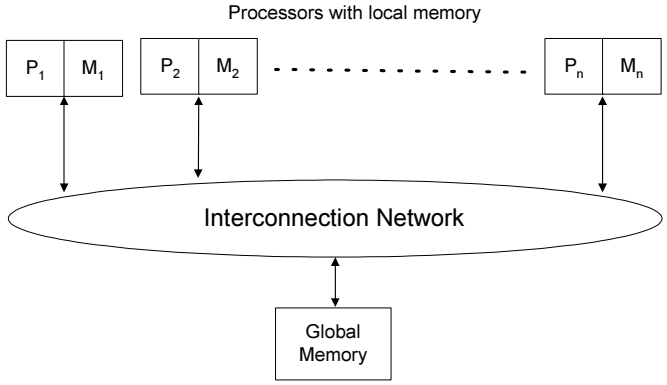


Fig. 9.7 : A global memory multiprocessor.

The way in which the local memory is used varies from one system to another. Normally, each processor accesses instructions and data form its own local memory. The interconnection network provides access from any processor to a global memory. This memory holds programs and data that can be shared among the processors, providing a means for interaction among the tasks performed by the processors. These tasks may be components of a single large application that has been divided into sub-tasks. They may also be independent tasks that require the use of some shared resources such as printers or magnetic disks. The relationship between the information stored in the local memories and that stored in the global memory.

The physical connection between processors and/or memory modules may take the form of a common bus.

The physical connection between processors and/or memory modules may take the form of a common bus. The processor themselves may have communications capabilities which allow them to be directly connected together. Alternatively, some sort of switching mechanism may be employed.

A crossbar switch is one which is able to connect a number of inputs with a number of outputs according to specified permitted combinations.

For example, a 2×2 crossbar switch has two inputs and two outputs. The switch will permit input i (i = 0 or 1) to be connected to output i, and also input i to be connected to output (i+1) mod 2.

Further, a broadcast can be achieved by permitting either input to be connected to both output simulate aneously. By increasing the number, N, of inputs and outputs the number of possible interconnections grows rapidly.

Another way of implementing an interconnection network is to construct a multistage switch in which each component is a crossbar switch. For example, four inputs can be connected to four outputs via a two-stage system in which the switches at each stage are 2×2 crossbars.

4.2. Speed-up and Efficiency

Speed-up and efficiency.

Let T_P be the execution time for a parallel program on p processors. We define the following terms;

- S_P, the speed-up ratio on p processors, is given by

$$S_p = T_0 / T_p$$

Where T_0 is the time for the fastest serial algorithm on a single processor. It should be noted that this definition of speed-up ratio compares a parallel algorithm with the fastest serial algorithm for a given problem. It measures the benefit to be gained by moving the application from a serial machine with one processor to a parallel machine with p identical processors. Even if the same algorithm is employed we normally expect the time taken by the parallel implementation executing on a single processor (T_1) t exceed the time taken by the serial implementation on the same processor (T_0) because of the overheads associated with running parallel processes.

4.3. Programming Languages

Programming Languages

The earliest machines had to be programmed in machine code, but it soon became clear that some higher level of abstraction was necessary. The development of FORTRAN (Formula Translator), ALGOL (Algorithmic language) and their derivatives. Closely reflect the von Neumann model; a FORTRAN program consists of a sequence of instructions which the program writer expects to be obeyed in sequence unless he explicitly includes a jump instruction to some other part of the code.

This has been accompanied by the development of software tools designed to assist the transfer of a sequential code to a parallel environment.

There are very few programming languages in common use that explicitly support parallelism. Rather, extensions have been made to existing languages, such as FORTRAN and Pascal (and even Basic), to provide appropriate facilities. Two languages which explicitly support concurrency, namely, Ada and occur, and also extensions to FORTRAN which are appropriate for multiprocessor systems.

In the case of array processors, languages such as CM FORTRAN and DAP FORTRAN have been developed to permit exploitation of the underlying architecture. For vector processors we can use a standard language, such as FORTRAN, and relay on a compiler to vectors the code, that is, produce object code which makes use of the hardware pipelines.

4.4. Performance Measurements

Performance Measurements

To compare the performance of various computer systems some means of measuring their speed is clearly required. We have already mentioned Mflops as a means of quantifying the speed of a floating-point unit. An

alternative performance measure frequently employed is mips (millions of instructions per second).

Manufacturers usually quote performance which refer to :

- peak speed, calculated by determining the number of instructions 9 or flops 0 that theoretically could be performed in the cycle time of the processor.
- Sustained speed, a measure of the speed likely to be achieved for a 'typical' application.

4.5. Processors and Processes

Processors and Processes

In the above discussion, we have used "multiprocessor" as a generic term for a machine which consists of a number of computing units, some form of interconnection and, possibly, a number of separate memory modules. Now, we need to differentiate between 'processors' and 'processes'. The former is a functional unit (hardware) and the latter is a combination of operations performed on data (software). On a uniprocessor we have only one process executing at any one time, but may have multiple processes time shared. On a multiprocessor system, a one-to-one correspondence exists between processors there will be P processes with one process per processor and no time sharing. On a local memory system, therefore, inter process communication will involve messages being sent along inter processor links.

4.6. Exercise

4.6.1. Multiple choice questions

a) The system performance and throughput capability system can be increased by increasing

i) clock speed
ii) number of instructions per second
iii) number of processor
iv) number of data.

b) FORTRAN stands for

i) Formula Translator
ii) Formula Transmission
iii) Function Translation
iv) none of the above.

c) A general MIMD configuration usually called

i) a multiprocessor
ii) a vector processor
iii) array processor
iv) none of the above.

4.6.2. Questions for short answers

a) How does multiprocessor system work ? What is the function of a crossbar switch?
b) Define speed up and efficiency.
c) What is multiprocessor?

4.6.3. Analytical question

a) Give a brief description on multiprocessor systems.