Parallel processing

Parallel processing is a method in computing of running two or more processors (CPUs) to handle separate parts of an overall task. Breaking up different parts of a task among multiple processors will help reduce the amount of time to run a program. Any system that has more than one CPU can perform parallel processing, as well as multi-core processors which are commonly found on computers today.

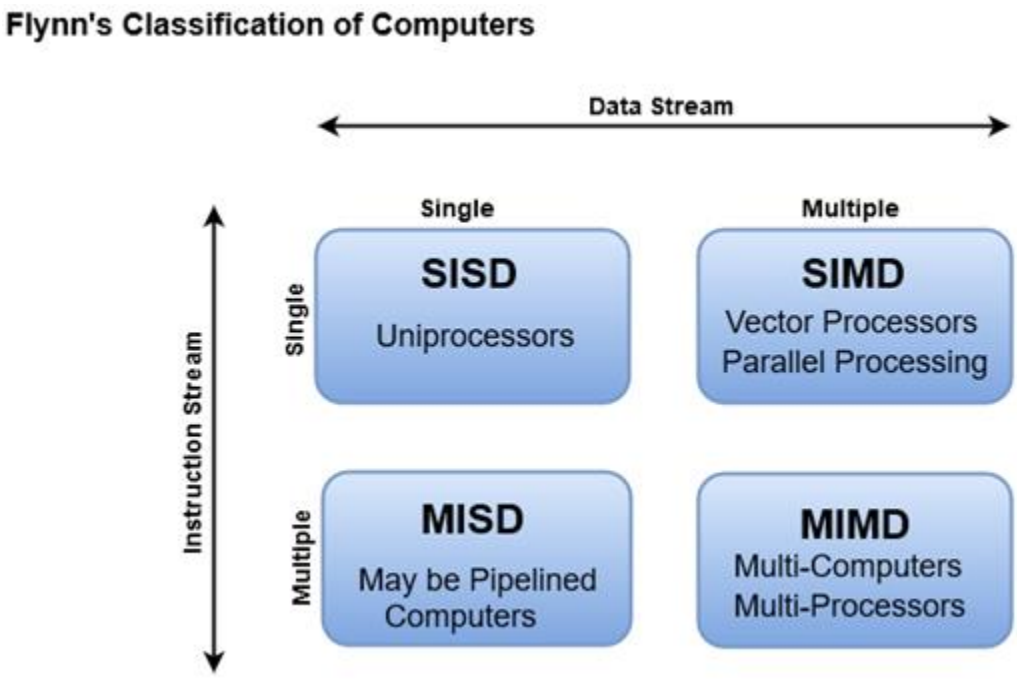
How parallel processing works

Typically a computer scientist will divide a complex task into multiple parts with a software tool and assign each part to a processor, then each processor will solve its part, and the data is reassembled by a software tool to read the solution or execute the task.

Typically each processor will operate normally and will perform operations in parallel as instructed, pulling data from the computer's memory. Processors will also rely on software to communicate with each other so they can stay in sync concerning changes in data values. Assuming all the processors remain in sync with one another, at the end of a task, software will fit all the data pieces together.

Computers without multiple processors can still be used in parallel processing if they are networked together to form a cluster.

Types of parallel processing



There are multiple types of parallel processing, two of the most commonly used types include SIMD and MIMD. SIMD, or single instruction multiple data, is a form of parallel processing in which a computer will have two or more processors follow the same instruction set while each processor handles different data. SIMD is typically used to analyze large data sets that are based on the same specified benchmarks.

MIMD, or multiple instruction multiple data, is another common form of parallel processing which each computer has two or more of its own processors and will get data from separate data streams.

Another, less used, type of parallel processing includes MISD, or multiple instruction single data, where each processor will use a different algorithm with the same input data.

Difference between Serial and parallel processing

Where parallel processing can complete multiple tasks using two or more processors, serial processing (also called sequential processing) will only complete one task at a time using one processor. If a computer needs to complete multiple assigned tasks, then it will complete one task at a time. Likewise, if a computer using serial processing needs to complete a complex task, then it will take longer compared to a parallel processor. Advantages

- Parallel computing saves time, allowing the execution of applications in a shorter wall-clock time.
- Solve Larger Problems in a short point of time.
- Compared to serial computing, parallel computing is much better suited for modeling, simulating and understanding complex, real-world phenomena.
- Throwing more resources at a task will shorten its time to completion, with potential cost savings. Parallel computers can be built from cheap, commodity components.
- Many problems are so large and/or complex that it is impractical or impossible to solve them on a single computer, especially given limited computer memory.
- You can do many things simultaneously by using multiple computing resources.
- Can using computer resources on the Wide Area Network(WAN) or even on the internet.

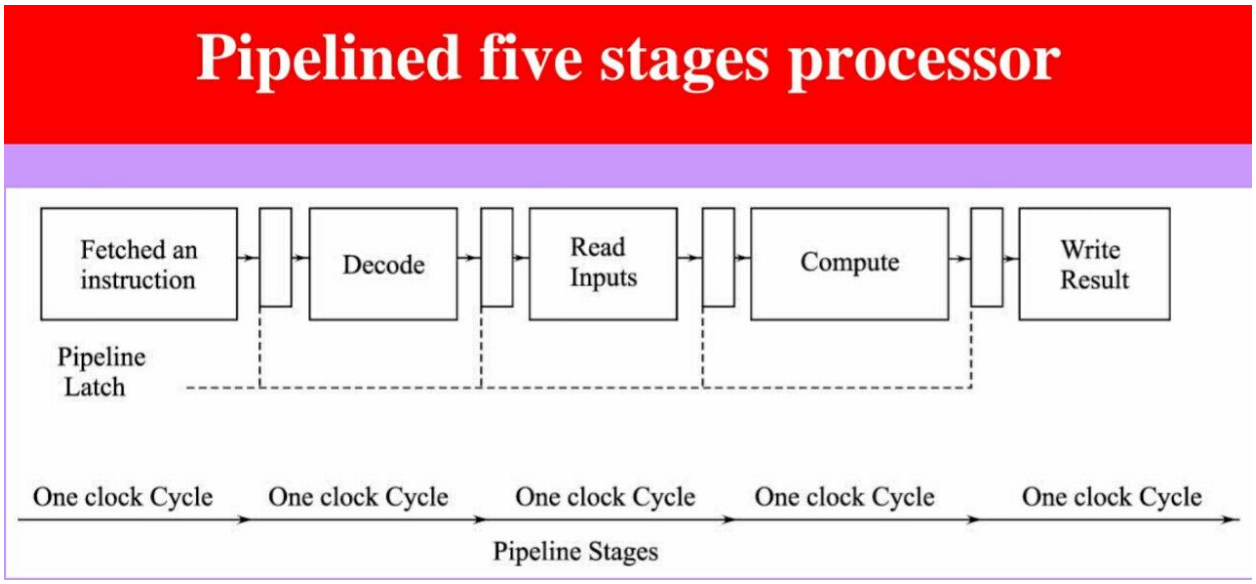
- It can help keep you organized. If you have Internet, then communication and social networking can be made easier.
- It has massive data storage and quick data computations.

Disadvantages

- Programming to target Parallel architecture is a bit difficult but with proper understanding and practice, you are good to go.
- The use of parallel computing lets you solve computationally and data-intensive problems using multicore processors, but, sometimes this effect on some of our control algorithm and does not give good results and this can also affect the convergence of the system due to the parallel option.
- The extra cost (i.e. increased execution time) incurred are due to data transfers, synchronization, communication, thread creation/destruction, etc. These costs can sometimes be quite large, and may actually exceed the gains due to parallelization.
- Various code tweaking has to be performed for different target architectures for improved performance.
- Better cooling technologies are required in case of clusters.
- Power consumption is huge by the multi-core architectures.
- Parallel solutions are harder to implement, they're harder to debug or prove correct, and they often perform worse than their serial counterparts due to communication and coordination overhead.

Pipelining

Pipelining is the **process** of accumulating instruction from the processor through a **pipeline**. It allows storing and executing instructions in an orderly **process**. It is also known as **pipeline processing**. **Pipelining** is a technique where multiple instructions are overlapped during execution.



The instruction pipelines

The ARM9EJ-S core uses a pipeline to increase the speed of the flow of instructions to the processor. This enables several operations to take place simultaneously, and the processing and memory systems to operate continuously.

A five-stage (five clock cycle) ARM state pipeline is used, consisting of Fetch, Decode, Execute, Memory, and Writeback stages. This is shown in Figure 1.1.

A six-stage (six clock cycle) pipeline is used in Jazelle state, consisting of Fetch, Jazelle/Decode (two clock cycles), Execute, Memory, and Writeback stages. This is shown in Figure 1.2.

Figure 1.1. Five-stage pipeline

During normal operation:

- one instruction is being fetched from memory
- the previous instruction is being decoded
- the instruction before that is being executed
- the instruction before that is performing data accesses (if applicable)
- the instruction before that is writing its data back to the register bank.

.

Arithmetic Pipeline

Arithmetic Pipelines are mostly used in high-speed computers. They are used to implement floating-point operations, multiplication of fixed-point numbers, and similar computations encountered in scientific problems.

To understand the concepts of arithmetic pipeline in a more convenient way, let us consider an example of a pipeline unit for floating-point addition and subtraction.

The inputs to the floating-point adder pipeline are two normalized floating-point binary numbers defined as:

$$X = A * 2^a = 0.9504 * 10^3$$
$$Y = B * 2^b = 0.8200 * 10^2$$

Where **A** and **B** are two fractions that represent the mantissa and **a** and **b** are the exponents.

The combined operation of floating-point addition and subtraction is divided into four segments. Each segment

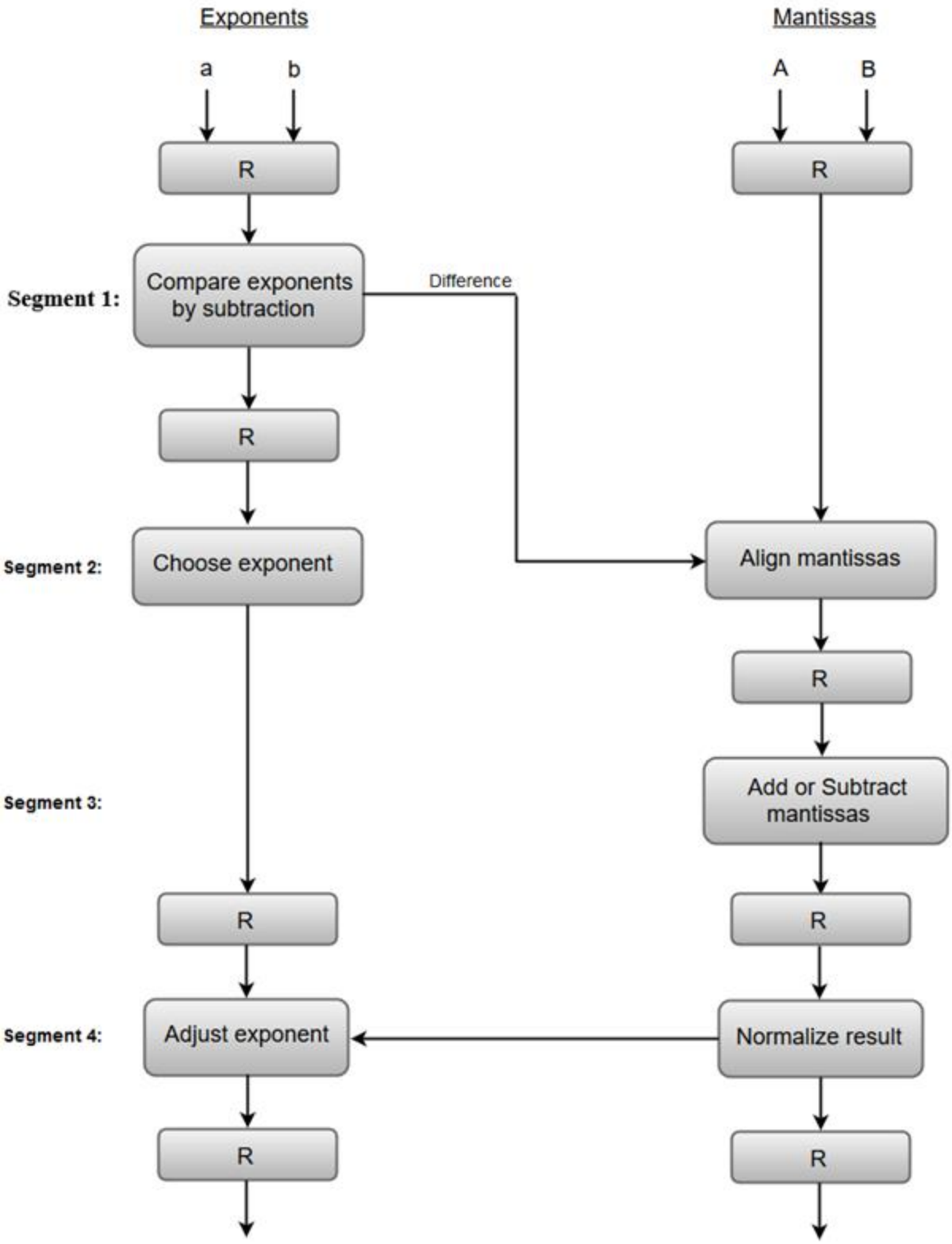
contains the corresponding suboperation to be performed in the given pipeline. The suboperations that are shown in the four segments are:

1. Compare the exponents by subtraction.
2. Align the mantissas.
3. Add or subtract the mantissas.
4. Normalize the result.

We will discuss each suboperation in a more detailed manner later in this section.

The following block diagram represents the suboperations performed in each segment of the pipeline.

Pipeline organization for floating point addition and subtraction:



1. Compare exponents by subtraction:

The exponents are compared by subtracting them to determine their difference. The larger exponent is chosen as the exponent of the result.

The difference of the exponents, i.e., **3** - **2** = **1** determines how many times the mantissa associated with the smaller exponent must be shifted to the right.

2. Align the mantissas:

The mantissa associated with the smaller exponent is shifted according to the difference of exponents determined in segment one.

$$X = 0.9504 * 10^3$$
$$Y = 0.08200 * 10^3$$

3. Add mantissas:

The two mantissas are added in segment three.

$$Z = X + Y = 1.0324 * 10^3$$

4. Normalize the result:

After normalization, the result is written as:

$$Z = 0.1324 * 10^4$$

RISC Pipelines

A RISC processor pipeline operates in much the same way, although the stages in the pipeline are different. While different processors have different numbers of steps, they are basically variations of these five, used in the MIPS R3000 processor:

1. fetch instructions from memory
2. read registers and decode the instruction
3. execute the instruction or calculate an address
4. access an operand in data memory

5. write the result into a register

If you glance back at the diagram of the laundry pipeline, you'll notice that although the washer finishes in half an hour, the dryer takes an extra ten minutes, and thus the wet clothes must wait ten minutes for the dryer to free up. Thus, the length of the pipeline is dependent on the length of the longest step. Because RISC instructions are simpler than those used in pre-RISC processors (now called CISC, or Complex Instruction Set Computer), they are more conducive to pipelining. While CISC instructions varied in length, RISC instructions are all the same length and can be fetched in a single operation. Ideally, each of the stages in a RISC processor pipeline should take 1 clock cycle so that the processor finishes an instruction each clock cycle and averages one cycle per

A **load delay** slot is an instruction which executes immediately after a **load** (of a register from memory) but does not see, and need not wait for, the result of the **load**. **Load delay** slots are very uncommon because **load delays** are highly unpredictable on modern hardware.

delayed branch. **delayed branch** A conditional **branch** instruction found in some RISC architectures that include pipelining. The effect is to execute one or more instructions following the conditional **branch** before the **branch** is taken.

A **multiprocessor** is a **computer** system with two or more central processing units (CPUs), with each one sharing the common main memory as well as the peripherals. This helps in simultaneous processing of programs. ... A good illustration of a **multiprocessor** is a single central tower attached to two **computer** systems.

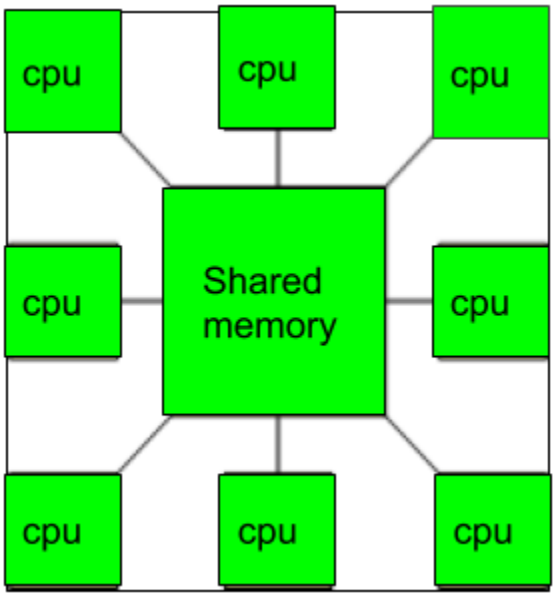
A **multiprocessor** is a **computer** system with two or more central processing units (CPUs), with each one sharing the common main memory as well as the peripherals. This helps in simultaneous processing of programs. ... A good illustration of a **multiprocessor** is a single central tower attached to two **computer** systems.

there are many types of multiprocessor systems:

- Loosely coupled **multiprocessor** system.
- Tightly coupled **multiprocessor** system.
- Homogeneous **multiprocessor** system.
- Heterogeneous **multiprocessor** system.
- Shared memory **multiprocessor** system.
- Distributed memory **multiprocessor** system.
- Uniform memory access (UMA) system.
- cc–NUMA system.

Introduction of Multiprocessor and Multicomputer

1. Multiprocessor:
A Multiprocessor is a computer system with two or more central processing units (CPUs) share full access to a common RAM. The main objective of using a multiprocessor is to boost the system's execution speed, with other objectives being fault tolerance and application matching.
There are two types of multiprocessors, one is called shared memory multiprocessor and another is distributed memory multiprocessor. In shared memory multiprocessors, all the CPUs shares the common memory but in a distributed memory multiprocessor, every CPU has its own private memory.



Applications of Multiprocessor –

1. As a uniprocessor, such as single instruction, single data stream (SISD).
2. As a multiprocessor, such as single instruction, multiple data stream (SIMD), which is usually used for vector processing.
3. Multiple series of instructions in a single perspective, such as multiple instruction, single data stream (MISD), which is used for describing hyper-threading or pipelined processors.
4. Inside a single system for executing multiple, individual series of instructions in multiple perspectives, such as multiple instruction, multiple data stream (MIMD).

Benefits of using a Multiprocessor –

- Enhanced performance.
- Multiple applications.
- Multi-tasking inside an application.
- High throughput and responsiveness.
- Hardware sharing among CPUs.

2. Multicomputer:

A multicomputer system is a computer system with multiple processors that are connected together to solve a problem. Each processor has its own memory and it is

Difference between multiprocessor and Multicomputer:

1. Multiprocessor is a system with two or more central processing units (CPUs) that is capable of performing multiple tasks where as a multicomputer is a system with multiple processors that are attached via an interconnection network to perform a computation task.
2. A multiprocessor system is a single computer that operates with multiple CPUs where as a multicomputer system is a cluster of computers that operate as a singular computer.
3. Construction of multicomputer is easier and cost effective than a multiprocessor.
4. In multiprocessor system, program tends to be easier where as in multicomputer system, program tends to be more difficult.
5. Multiprocessor supports parallel computing, Multicomputer supports distributed computing.