UNIT 16 ADVANCED ARCHITECTURES

Structure **Page Nos.**

16.0 INTRODUCTION

The previous Units of this course discuss about the basic computer architecture of a computer system, including the assembly language. Architecture design is the first step in life cycle of a processor. It is a very crucial step as the performance of processor majorly depends on the design chosen. For instance, if you choose a non-pipelined architecture for your processor, you are compromising its performance for simplicity. On the other hand, if you choose an architecture involving multiple cores, you can increase your processor performance exponentially but at the same time its complexityincreases drastically. So, choosing an architecture that suits our application is a must. For that you should have an understanding of various types of architectures and their implementations in detail.

In this Unit, we will discuss some advanced architectures and design methodologies used to achieve higher performance of a computer system.

16.1 OBJECTIVES

After going through this Unit, you will be able to:

- define the concept of parallelism in both uniprocessor and multi-processor system.
- define various techniques used to implement pipelining in a processor.
- explain the concept of multiprocessor systems and interrelated details.
- define how processors communicate with each other.
- define the concept of Cache Coherence and
- define various types of multi-core processors.

16.2 NEED OF ADVANCED ARCHITECTURES AND PARALLEL PROCESSING

The fast-paced IT industry demands high-end architectural designs to support its high-performance applications. To meet these needs computer architectures are going through numerous changes constantly. One such implementation is parallel processing which is used to achieve higher speeds.

Parallel Processing includes a set of techniques that can be used to increase processing speed and latency, which is defined for the data rate of data transfer, of a computational system. These techniques enable simultaneous processing of data as opposed to the conventional sequential processing. Due to parallel execution of instructions, there is a significant increase in throughput. *Throughput* is the measure of computation and is defined in terms of number of instructions that can be executed by a given processor in a given interval of time. Increase in throughput implies to increase in speed of operation. To this extent it must benoted that the need of parallel processing is to increase the performance of a system. The techniques involved in inducing parallel processing in a system widely vary in methodology and resources used. However, the aim of all these techniquesis to process data concurrently. For example, in a processor, one instruction can be fetched from memory and another instruction can be executed by the ALU in the same clock cycle. Another example can be a system using two processors for parallel execution of instructions. In the former example, the Processor is using Pipelining as a technique to speed up the execution of instructions. We will discuss pipelining in the next sections in detail. The later example of two processors is used to achieve parallel processing. Here, performance of the system increases at the expense of increased complexity and cost.

16.3 PARALLELISM IN UNI-PROCESSOR SYSTEMS

Parallel execution of several program can be done on a single processor system. Such parallelism can be implemented using the hardware as well as software. In this unit, we will focus only on the hardware related parallelism in a uni-processor system.

As far as hardware-based parallelism is concerned several techniques are used on uni-processor system to increase the throughout of instruction execution. Some of these techniques are:

- Several processors contain multiple units to perform various arithmetic functions, such as multiple adder circuits unit as designed in Block 1. These units' speedup the execution of various functions that can be done in parallel.
- Using the memory organizations, such as interleaved memory, to speed up the data access operations. In addition, the processor operations can be overlapped with memory operations, for example, 8086 micro-processor
- Use of pipelining in the processor. Which is explained in detailed in this section.

The concept of pipelining is one of the major aspects of Parallelism in Uniprocessors, let us first answer the question "What is Pipelining?".

Pipelining is a technique in which we divide the whole task into subtasks and execute them concurrently. Each subtask is processed in different segment of the processor. These segments are interconnected to each other in such a way that the result of one segment is passed to another segment. Output is obtained after the data is processed through all the segments. The characteristic feature of pipelining is that several processes can be running in different segments at the same time. Typical related example of pipelining can be an assembly linein industries, such as automobile manufacturing, which fabricate automobile in a step-by-step manner in different segments. In the following sub-section, the concepts of arithmetic and instruction pipeline are explained.

16.3.1 Arithmetic Pipeline

In Arithmetic Pipelining an arithmetic operation is divided into a sequence of segments and computations of several arithmetic operationscan be done concurrently. Therefore, in this technique, there may be a possibility that a segment has produced data for the next segment, which is still processing the data of earlier operation. This will result in overlapping of data in different segments. To overcome this problem,you can use registers between segments to store data while next segment is still computing. Adding registers between stages can increase complexity but this step improves the reliability of the system.

Arithmetic pipelining is preferred in those systems in which same operation is to be performed on different data sets multiple times. The arithmetic pipeline can be used in computers used for high-end scientific computations to implement the floating point number arithmetic etc.

To describe Arithmetic Pipelining, let us consider apipeline that adds two floating-pointnumbers. Let us consider two numbers represented in floating point format.

$$A = X \times 10^{x}$$

$$B = Y \times 10^{y}$$

Here, X, Y are the mantissa of numbers A, B respectively and x,y are the exponents of A and B respectively. In order to find the sum of the two floating point numbers A and B, the following sequence of steps are required to be computed:

Step1: Compare the exponents of the given floating point numbers A and B, i.e. compare the values of x and y.

Step2: Align the mantissa of the number having smaller exponent. It may be noted that this process may result in unnormalized mantissa representation.

Step3: Perform the addition of mantissas to obtain resultant mantissa, while exponent of the bigger number is chosen as the exponent of the result.

Step4: If the resultant mantissa is not normalized, then normalize the result.

The following example explains the process of addition of floating-point numbers.

Example: Add the following two floating point numbers using the process as explained above:

$A = 0.5565 \times 10^{4}$ and $B = 0.166 \times 10^{3}$

Step1: The exponent of A, which is 4, is greater than that of exponent of B, which is 3.

Step2: In this step, you should align the mantissa of thesmaller numbers by modifying the B as follows (please note that A remains unchanged). Please note after alignment exponent of both A and B is same, however, the mantissa of B is unnormalized.

$B = 0.0166 \times 10^4$

Step3: Add the two numbers as follows:

$$Result = A + B = 0.5565 \times 10^4 + 0.0166 \times 10^4 = 0.5731 \times 10^4$$

Step4: Since the result is already normalized, this step will not perform any operation. Thus, the final result of the addition operation is 0.5731×10^4.

Registers are present in between all these steps to store intermediate values from one step to another step and prevent overlapping of data. The following diagram illustrates this process of addition or subtraction of two floating point numbers.

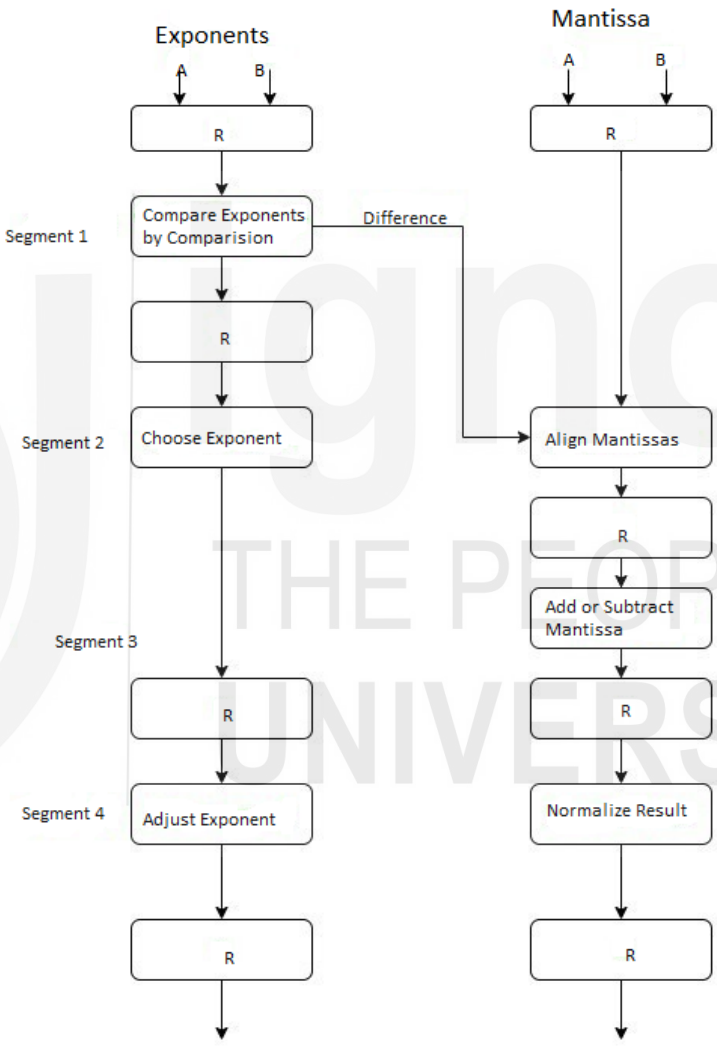

Figure 16.1: An arithmetic pipeline organization with intermediate registers for addition or subtraction of two floating point numbers.

16.3.2 Instruction Pipeline

In a processor when instructions are executed in various pipeline stages concurrently, then it is called instruction pipelining. The execution of an instruction is performed in asequence of stages. For example, in a 3-stage pipelined processor every instruction undergoes three stages namely Fetch, Decode, Execute.

Fetch: In the fetch stage, the instruction is fetched from instruction memory and stored in Instruction Register (IR).

Decode: In this stage the instruction is decoded for information like operation to be performed, source operand address, destination operand address etc., the source operand and destination operand may be stored in temporary registers for further processing.

Execute: In this stage the ALU performs the operation specified by the instruction on the operands in the temporary registers.

In the pipelined processor as stated above, when the first instruction is in decode stage the second instruction enters into fetch stage. When first instruction enters execute stage, second instruction enters decode stage and third instruction enters fetch stage. The process continues until all the instructions are out of the execution stage.

Assuming that one stage is performed in one clock cycle, then this processor will take**(k+n-1)** clock cycles to execute **n** instructions in a **k** stage pipeline. In the present case, k= 3, as we have used athree-stageinstruction pipeline. Thus, using this instruction pipeline **n** instructions, in an ideal condition, would be executed in **n+2** clock cycle.

If you are executing 100 instructions, then this processor will take 102 clock cycles to execute all of them. A Non-pipelined processor would have taken 300 clock cycles to complete the same task i.e., n×k.

We could see that there is a speedup ratio (S) $=\frac{(n\times k)t}{(k+n-1)t}$

Instruction 1	**FE**	**DE**	**EX**				
Instruction 2		**FE**	**DE**	**EX**			
Instruction 3			**FE**	**DE**	**EX**		
Instruction 4				**FE**	**DE**	**EX**	
Instruction 5					**FE**	**DE**	**EX**

Figure 16.2: An Instruction pipeline demonstrating execution of five instructions

The computation of speed up due to pipeline though looks very promising, however, the pipelined execution suffers from several problems. The first problem is due to limitation of resources of the processing unit. For example, the system bus is one of the resources required by several units.The second problem may be related to data dependencies among consecutive instructions, for example, an instruction may produce a result, which may be required by the immediate next instruction. This may sometimes may cause delay in execution of this immediate next instruction. Finally, in general, the decision to take a jump in conditional jump instructions is known only when the EX phase of that instruction is performed. This may result in emptying the pipeline. For example, consider instruction 1 is a conditional jump instruction, which causes jump to instruction 4 in case the condition is TRUE. The pipeline will execute as shown in the following diagram, assuming that the condition is evaluand to be TRUE.

Conditional Jump Instruction 1	**FE**	**DE**	**EX**				
Instruction 2		**FE**	**DE**	**-**			
Instruction 3			**FE**	**-**	**-**		
Instruction 4				**FE**	**DE**	**EX**	
Instruction 5					**FE**	**DE**	**EX**

Figure 16.3: An Instruction pipeline with conditional jump

Please note that the instruction 2 and instruction 3 are not to be executed, still they will be fetched. There a number of methods such as branch prediction, which can be used to handle such problem. A detailed discussion on these problems are beyond the scope of this Unit.

☞ Check Your Progress-1:

1) **State True or False**

i) A Non-pipelined processor is faster than a pipelined processor ☐

ii) Implementing parallelism in a system may lead to increased complexityof the system ☐

iii) Throughput is the measure of area of the chip ☐

iv) Modern day processors relay mostly on sequential data processing to improve their performance ☐

v) A Non-pipelined processor takes $n \times t_n$ clock cycles to complete n tasks with a clock period of t_n secs. ☐

2) Calculate the number of clocks cycles a processor takes to complete 180 instructions in a 7-stage pipeline.

……

……

……

……

3) Where can you use Arithmetic Pipelining?

……

……

……

……

16.4 PARALLELISM THROUGH HARDWARE AND SOFTWARE

As we know parallelism is the concept of processing multiple tasks concurrently. If this parallelism is achieved through hardware, then it is called Hardware parallelism. Similarly, if parallelism is achieved through software, then it is called Software parallelism.

Hardware Parallelism:

- Hardware parallelism is implemented in a system at the architectural level by implementing hardware multiplicity or by modifying machine architecture.
- While implementing hardware parallelism one should keep in mind that it is a tradeoff between cost and performance.
- For example, using multiple adders can speed up a system which is initially working with single adder. But there is an overhead of cost of extra adders implemented in the system. Depending on the application you should choose between cost and performance.

Software Parallelism:

- Software parallelism is achieved by modifying programs for data dependencies and control.
- Parallelism in software varies during the time of execution.

In the next section we discuss vector processing, which propose software instructions to exploit parallelism. This would be followed by a discussion of array processor, which is a hardware to exploit the parallelism.

16.4.1 Vector Processing

The capabilities of a conventional computer are limited to computations that are simple and not as complex as vector or matrix arithmetic. But some modern-day applications need more complex calculations to be performed on huge amount of data. In this scenario, performing computations on scalars one by one is not recommended. To overcome this limitation, computers with vector processing are preferred to do faster vector calculations. There are many scientific problems and real-time computations that could take many weeks on a conventional computer to complete. Vector processers do the same problem in a much faster way. But how do they do that? The following description and exampleattempt to answer this question.

Instruction sets in vector processors contain instructions that operate on vector data, such as one-dimensionalarray. In scalar processors the instructions operate on a single data element at a time.

A scalar subtraction operation can be represented asC= A-B, where the scalar value B is subtracted from scalar value A in a single step.

However, if both A and B are vector operands, you may perform the subtraction operation, as shown below:

$C_i = A_i - B_i$, where i=1,2,3, 4,…*n* and *n* is the number of elements in the vector

If you follow the above procedure, then you will subtract vector B from vector A in *n* steps.

In vector processing vectors are subtracted in a single step. This procedure saves (n-1) clock cycles, which makes a huge difference when processing large arrays of data.

Vector instructions and scalar instructions are specified in a different way. For example, a typical scalar instruction having three operands can be defined as follows:

Scalar instruction format:

Opcode	Address of source 1	Address of source 2	Address of destination

Whereas, a typical vector instruction would require to specify the base addresses of the operands and the length of the vector, which specifies the number of elements in those vectors. A typical vector instruction can be represented as follows:

Vector Instruction Format:

Vector Opcode	Base address of source operand1	Base address of source operand2	Base address of destination operand	Length of vector

Example: SUB X,Y,Z,10

Here opcode is SUB, this specifies that the operation to be performed is subtraction.Base address of source operands are X and Y respectively and the Base address of destination operand is Z. The length of vector is 10.

The scalar version of the program may be written as follows:

for i = 1 to 10 SUB X[i], Y[i], Z[i]

The advantage of the vector instruction over scalar instruction is that the scalar version of instruction would require repeated fetch and decode of the subtraction instruction, whereas, the vector instruction would be fetched and decoded only once. The efficiency of execution of vector instructions can be further enhanced by using an array processor, which is discussed next.

16.4.2 Array Processing

Array processing is a technique in which arithmetic operations are done on large arrays of data. Array processing also works on vectors of data, as explained in the previous section. In general, to support array processing array processors are used in a computer.Array processors perform operations on vectors, but they operate on large data sets at the same time.

Array processors are of two types:

1) Attached array processors

2) SIMD array processors

Let us discuss these two array processors in detail:

1) Attached Array Processors:

Attached array processors are used to aid host computer as an auxiliary processor or co-processor to perform array operations. Host computer can be a general-purpose computer that deals with simple computations and operations. When this general-purpose computer needs to operate on complex data like arrays, an array processor comes into play.

An array processor is interfaced with its host computer as follows:

1) I/O interface with host.

2) Interfaced to main memory of host.

Since array processor is attached to host computer through I/O interface, host computer treats it as a peripheral device.

2) SIMD array processors:

SIMD stands for Single Instruction Multiple Data,i.e. in a single instruction they process multiple data. This is achieved by having multiple processing elements working in parallel.All the processing elements are controlled using a single controller. Thus, a single instruction is sent to all the processing units, which are fed with different data by their local memory.

Every processing unit contains three elements:

a) Arithmetic and Logical Unit (ALU)

b) Floating point arithmetic unit

c) Working registers

Control unit stores the instructions while processing units store operands.

Example:A vector operation $C_i = A_i \times B_i$,where$i=1,2,3\ldots$is to be performed using SIMD Array processor.

The array processor will perform the following steps, for execution of the instruction as given above:

Step1:Control unit checks whether the data is scalar or vector.If data is scalar, then control unit performs the operations directly in the master processing unit.If data is vector it sends data to Processing Units (PUs) and the following steps are performed:

Step2: Control unit sends data to processing units such that PU_1 stores operands A_1 and B_1, PU_2 stores operands A_2 and B_2 and so on.

Step3: Finally, control unit send instruction to multiply the operands to all the PUs.Thus, all the data is processed concurrently in the array processors.

Depending on the length of the vector, the control unit decides the number of PUs that should be activated. For example, if the length of vector to be processed is 32, then only 32 PUs are activated, and all other PUs are masked. In case, the length of a vector is more than the number of PUs, then all the PUs are activated and control unit uses these units as per requirements.

☞Check Your Progress – 2

1) **State True or False**

	T/F
i) An Array processor performs operations on vector data	☐
ii) SIMDarray processors contains a single processing unit and multiple control	☐
iii) Masking is a technique of activating Processor Units (PU) in SMID Array Processor.	☐
iv) Throughput of Vector Processor is higher than Scalar Processor	☐

2) What are the components present in Processor Unit in an SIMD Array Processor?

……

……

……

……

3) Where can you use Vector Processing?

……

……

……

……

16.5 MULTI PROCESSORS

Multiprocessor systems refer to those systems that use multiple processors.These processors execute multiple tasks by sharing them on multiple processors simultaneously. You can think of multiprocessor as a system consisting of different processing units working together.

At the operating system level, multiprocessing can be referred as processing multiple processes simultaneously, with each process or task being executed by different cores. This is different from single processor executionflow.Major difference between multiprocessor and single processor flow comes from the fact that, in multiprocessor systems single task is executed by multiple cores, whereas, in single processor system multiple tasks are executed concurrently over a single core. This is called as multi-tasking.Sometimes multi-tasking is being confused with multi-processor, it may be noted that multi-tasking uses single processor but switches among the tasks on completion of a time slice allotted to a specific task. The Figure 16.4 describes the use of multi processors (CPU) while using the common shared memory.

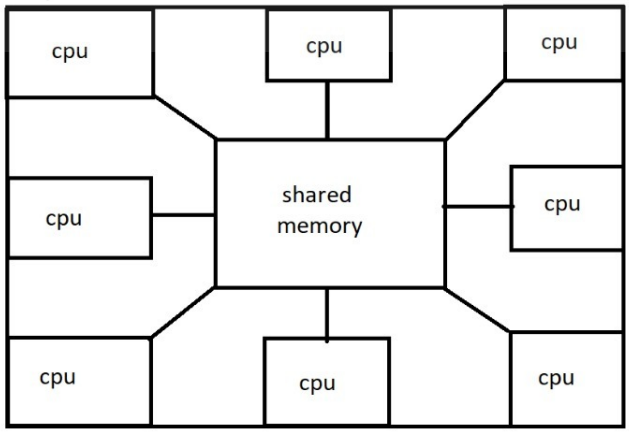

Figure 16.4: Multiprocessors with common shared memory

16.5.1 Characteristics of Multiprocessor

Multi-processor is a tightly coupled system with more than one CPU sharing same memory and input-output resources. Some of the major characteristics of multiprocessor are as discussed below:

- Processors used in multiprocessor system can be eitherCentralProcessing Units (CPUs) or Input-Output Processors (IOPs).
- They are categorized as Multiple Instruction Multiple Data(MIMD) systems.
- Major difference between a multiprocessor and a multicomputer system is, in multi computer system are connected with one another to form a network and they may or may not communicateinformation among them.
- A multiprocessor system may be designed to have a singlemaster Operating System. This operating system may be responsible for communication between the processors, other peripheralsand shared memory of the system.
- Multiprocessing can improve thereliability of the system such that when a failure happens, it affects only that processor, therefore, it has very less impact on rest of the processors.
- If one of the processors has failed, then some other processor can take up the duties of the faulty processor.
- Multiprocessing will enhance performance by breaking down a program into parallel executable tasks. Multiprocessors are flexible,i.e. the user can explicitly declare which processes are to be executed in parallel.
- Other effective way to improve performance of the system is to use a compiler which can identify parallelism in a process automatically.
- Compiler also tries to debug for data dependency in the program, which may cause issues while executing a task in parallel.
- Two threads of a task, which use different data can run concurrently.
- Multiprocessor systems can be implemented as ~~barely~~ loosely coupled systems as well.Each element of a processor in ~~barely~~ loosely coupled system has its own independent local memory.

16.5.2 Inter-Connection Structures

Every processor in a multi-processor system has a set of components, namely:

- CPU
- Shared Memory
- I/O

These components should communicate with each other for proper execution of programs. These components communicate among themselves through some interconnect paths;these paths which connects these modules or components are called as interconnection structures.

Let’s discuss various interconnection structures in the detail.

Multi-port memory Organization

The multiport memory organization is illustrated in Figure 16.5. The multi-core processors have been using the shared memory resource modules (MM1, MM2, MM3, MM4) with the bus interconnection, which is used as a path for communicating among them. In this interconnection structure every CPU can communicate with every memory module. The advantage of such kind of connection is that, in case every CPU is trying to access a different memory module, they will be allowed access in parallel. However, their can be conflict, if to processor are trying to access the same memory module at the same time. A detailed discussion on this topic is beyond the scope of this Unit.

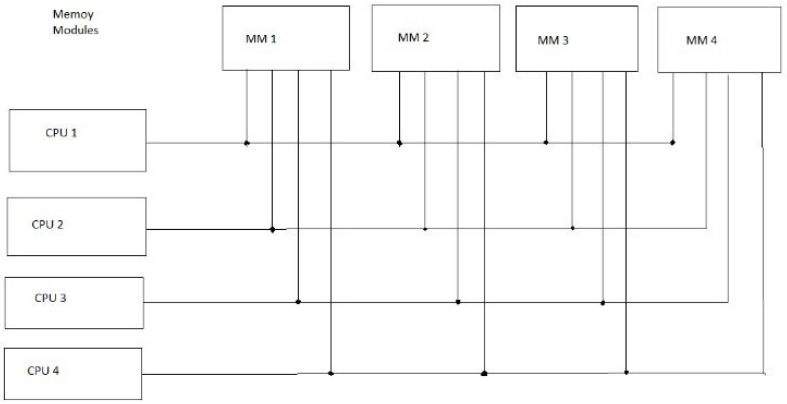

Figure 16.5: Multi-port memory Organization

Bus Interconnection Structure

A bus is used as a communication path for connecting two or more processors or devices. Bus is ashared transmission medium; this is a major advantage of bus interconnects. Bus has already been explained in the Block 2 of this course. In this section, let us discuss the serial arbitration procedure used in the buses, in the context of priority scheme for bus allocation. Figure 16.6 shows the serial arbitration process.

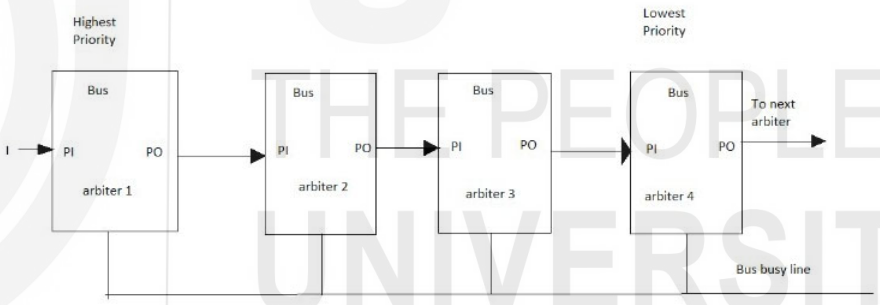

Figure 16.6: Serial Arbitration Procedure

In the above figure, Serial Arbitration procedure is illustrated. The arbiters are sharing a common bus busy line which is synchronized to maintain the synchronous communication. The input (l) is serially transmitted through the arbiters with the enabling of bus busy line. It might depend on the clock edge occurrence of bus busy line to transmit the input data through the arbiters.

16.6 INTER PROCESSOR COMMUNICATION AND SYNCHRONIZATION

In this section we will be discussing about Inter process communication facilities. Various applications of inter-process and inter-thread communication facilities, which uses data transfer are:

- Pipes (named, dynamic – shell or process generated)

- TCP/IP socket communication (named, dynamic - loop back interface or network interface)
- D-Bus is an IPC mechanism offering one to many broadcast and subscription facilities between processes.
- Shared memory
- Between Processes
- Between Threads (global memory)

Figure 16.7 shows and Figure 16.8 shows two simple mechanisms of inter-processor communication: message passing and shared memory.

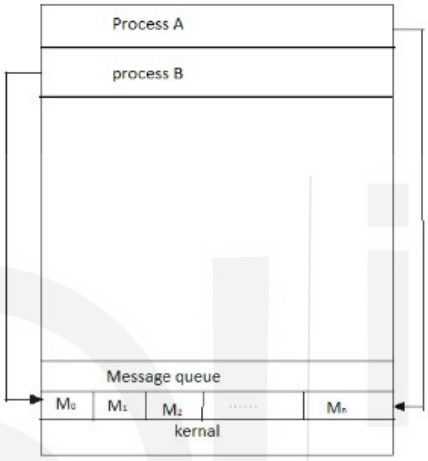

Figure 16.7: Message Passing

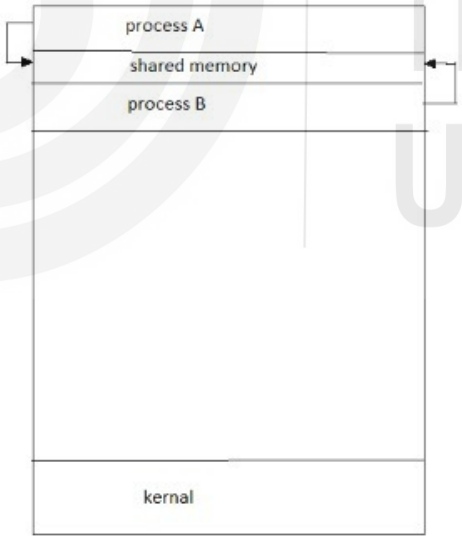

Figure 16.8: Shared Memory

The above figure illustrates the Inter-processor communication in two possible ways. The first one is with respect to two processes A and B trying to pass messages to the message queue. It depends on the algorithm used, which decides on which process gets to pass the message first.

The second figure indicates the two processes A and B trying to use the common memory and ultimately depends on the algorithm designed, which decides how the processes share the memory.

When two processes need multiple shared resources at the same time in order to proceed further a **Deadlock** condition occurs.

To understand this condition, let us consider a scenario in which Thread X is expecting data from Thread Y for further processing and Thread Y is expecting data from Thread X for further processing of data. This scenario is called deadlock and no further processing can be done between the two threads.

Operating System provides synchronization and communications between processes sharing resources and prevents them from facing potential Inter process communication problems.

16.7 CACHE COHERENCE

In multi-processor system, while processing the shared data, various processors can store this shared data in multiple local caches for faster local processing. Therefore, there is a chance of having caches with incoherent data. This can lead to non-uniformity in shared resources. This problem is predominant in the case of multi-processors that use multiple processors and have shared data.

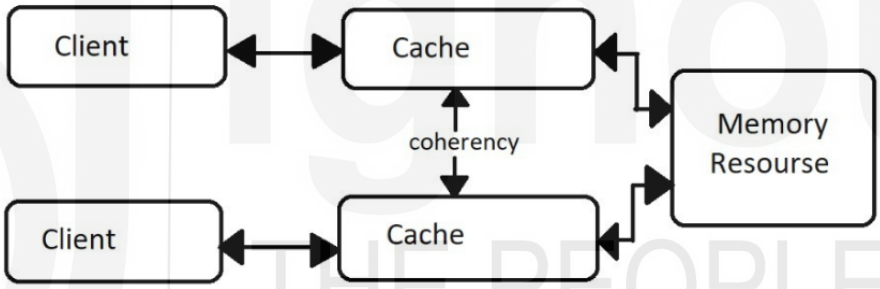

Figure 16.9: Cache Coherence

In the Figure 16.9, there are two client processors having their own local cache memory, which are initiallycoherent. Next, one of the client processors updatesits cache, which in turn changes the data in the shared memory. The other client processor should also update its cache memory to reflect the change made by the first client processor. This is how cache coherence works. However, what happens if the other processor is not able to update its cache. In this case, cache coherence is not present. Thus, the update of data made by one processor though is reflected in its cache and shared memory, but the other client processor has no clue about it. There will be data conflict between both these client processors. To overcome such conflicts and synchronize data between multiple caches, Cache Coherence protocols are used. The detailed discussion on these protocols is beyond the scope of this Unit.

16.8 MULTI-CORE PROCESSORS

Multi-core processor is adesign in which a number of cores are integrated on a single chip. These cores can be treated as processing units. They can process data and execute programs, similar to that of any other processor. It can be considered as a system having multiple processors. This system executes normal instructions such as add, subtract, and branch instructions. But these instructions can be executed on various cores at the same time. This results in drastic increase in speed by giving scope to multithreading and parallel computing.

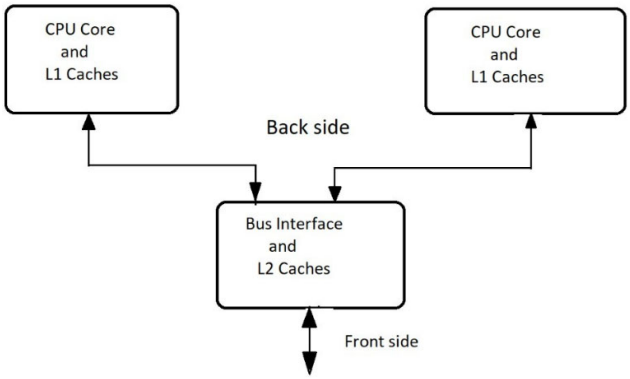

Figure 16.10: Multi-core Processor block diagram

Figure 16.10 shows the basic architecture of cores in a multi-core processor. Applications of Multi-core processors are in various domains such as embedded systems, networks, Digital Signal Processing (DSP) and other domains.

Performance of Multi-core processors depends mostly on algorithms and methodologies used to implement them. In general, a good multi-core processor requires a strong support from its operating system so that all the cores of the processor are efficiently used. This Software parallelization is one of the areas of the present research.

☞ Check Your Progress – 3

1. Explain Inter-processor Communication with various techniques.
   ……
   ……
   ……
   ……

2. Explain about Cache Coherence
   ……
   ……
   ……
   ……

3. Give some examples on Multi core processors
   ……
   ……
   ……
   ……

16.9 SUMMARY

This Unit provides a basic introduction pf the concepts of pipelining and multiprocessor. The two pipelining techniques discussed in this Unit are – arithmetic pipeline and instruction pipeline. Various Parallelism techniques such as Vector and array processing are also discussed in this unit, and further topics like Multiprocessors are demonstrated with diagrams and explained in brief.

The information given on various topics such as pipelining, both arithmetic and Instruction pipeline, is introductoryand can be supplemented with additional reading. In fact, a course in an area of computer must be supplemented by further reading to keep your knowledge up to date, as the computer world is changing with leaps and bounds. In addition to further readings the student is advised to study several Indian Journals on computers to enhance her/hisknowledge.

16.10 SOLUTIONS

Check your progress 1:

1.

i) False
ii) True
iii) False
iv) False
v) True

2. Number of stages in Pipeline is 'k' =7
Number of instructions is 'n' = 180
If each clock cycle is tp
Then the number of clock cycles taken by the processer to process 180 tasks in a 7-segment pipeline is = (k+n-1)tp
= (7+(180-1))tp
=186tp
Number of clock cycles for pipelined processor = 186

3.

- In high-speed Computers
- For doing floating point arithmetic
- For processing data of scientific problems

Check your progress – 2

1.

i) True
ii) False
iii) False
iv) True

2. Processing Unit contains the following components:

- ALU
- Floating Point Unit
- Registers

3. Vector processing is used in following fields:

- Weather forecasting
- Artificial Intelligence (AI)
- Image Processing
- Healthcare and diagnosis

Check your progress- 3

1. The inter processor communication can be achieved either through a shared memory area or through a queue. However, in both the cases software algorithms must control the sequence and synchronization of such communications.

2. In the case of multi-processors having shared memory and local caches, it is possible that a shared data item may be present in several local cache of the processors. In case, any of these processor changes the value of the shared data in it local cache, then it should result in update of this data in the shared memory and all other local caches. The coherence of cache is a required feature to keep the computation error free.

3. These days due to ULSI technologies most of the processors are multi-core processors. The Intel latest processors, AMD processor, processors of mobiles etc. all have multi-cores.