UNIT

Reduced Instruction Set Computer, Pipeline and Vector Processing, Multi Processors

SYLLABUS

Reduced Instruction Set Computer: CISC Characteristics, RISC Characteristics.

Pipeline and Vector Processing: Parallel Processing, Pipelining, Arithmetic Pipeline, Instruction Pipeline, RISC Pipeline, Vector Processing, Array Processor.

Multi Processors: Characteristics of Multiprocessors, Interconnection Structures, Interprocessor Arbitration, Interprocessor Communication and Synchronization, Cache Coherence.

LEARNING OBJECTIVES

- ✓ RISC and CISC Characteristics
- ✓ Types of Parallel Processors
- ✓ Classification of Pipeline Processors
- ✓ Three and Four Segment Instruction Pipelines
- ✓ Vector Processing and its Operations
- ✓ Array Processors and its Types
- ✓ Various forms for Establishing an Interconnection Network
- ✓ Different Types of Interprocess Arbitration and Interprocess Communication
- ✓ Cache Coherence Approaches and Solutions.

INTRODUCTION

The RISC (Reduced Instruction Set Computer) processor instruction sets are smaller in size and includes nearly 100 instructjons within it. Whereas, CISC (Complex Instruction Set Computing) is another design strategy which consists of full set of computer instructions and includes 120 to 350 instructions.

A method that is used to decompose a sequential process into sub operations is called pipelining. In this method, all the subprocesses will be executed in a separate segment that can operate along with the remaining segments. Thus, when the binary information flows through several processing segments, it is said to be in pipeline. Hence in general mathematics, the computations are implemented using high-level matrix and vector notations. The execution of these vector notations are often referred to as vector processing. The computers that combine the features of vector processing along with the floating-point computations are generally referred to as super computers. However,an array processor refers to a special category of processors which was developed to perform computations on larger volumes (array) of data.

In a multiprocessor environment, processors achieve parallelism by doing concurrent processing. Whereas, interprocessor arbitration can be described as a bus controller or arbitration unit that forms connections. Moreover, cache coherence refers to situation wherein multiple cached copies of the shared data are maintained such that all copies have the same value.

5.1 REDUCED INSTRUCTION SET COMPUTER: CISC CHARACTERISTICS, RISC CHARACTERISTICS

Q1. Write a short note on RISC and CISC.

OR

What are RISC instructions? Explain.

(Refer Only Topic: RISC)

April/May-23(R18), Q5(b)

Answer :

RISC

The RISC (Reduced Instruction Set Computer) processor instruction sets are smaller in size and includes nearly 100 instructions within it. These instructions must carryout wide range of functions and must be able to control any type of task. Moreover, all the instructions must be capable enough to get executed within one clock cycle. The designer must ensure that, the size of the instruction must be carefully chosen.

The RISC CPU is capable of accomplishing all the necessary functions needed by the task. However, with respect to CPU, the RISC instruction set contains few basic types of instructions such as,

1. **Data Move:** This instruction carries out operations like load, move, store and register.
2. **ALU:** This instruction carries out operations like arithmetic, logic and shift.
3. **Branch:** This instruction carries out operation like jump.

Apart from this, the CPU also contains 12 instructions that deals with exceptions. The RISC processor also includes new classes of instructions such as MIPS 4000 CPU that comprises of coprocessor and 11 instructions within its instruction set.

RISC CPUs are usually programmed to operate directly on one or more datatypes such as integer and floating points. Perhaps, there is a possibility that, the CPU can operate on different precisions for a given datatype. However, the CPU cannot manipulate character strings and other datatypes directly as it does not include instructions. But, they can be certainly programmed in order to deal with these datatypes using existing instructions within the instruction set. Due to lack of dedicated string instructions, these data operations are considered as time consuming. However, by pruning them out from the instruction set, the overall system performance can be improved.

Thus, RISC processor facilitates various instruction formats and irrespective of utilization of specific format, each and every instruction should posses equal number of bits.

CISC

CISC (Complex Instruction Set Computer) is another design strategy which consists of full set of computer instructions. These instructions are responsible for providing the required capabilities efficiently. In this type of CPU design strategy, a single instruction can execute multiple low-level operations. Moreover, the instruction-set is even capable of executing multi-step operations or addressing modes within single instruction. The instruction-set consists of 120 to 350 instructions.

Q2. Differentiate between RISC and CISC characteristics.

Model Paper-I, Q10(a)

OR

What are CISC characteristics?

(Refer Only Topic: CISC)

Answer :

April/May-23(R18), Q1(j)

The differences between RISC and CISC characteristics are as follows,

	RISC		CISC
1.	In RISC, simple instructions consume only one cycle to execute and an average CPI (cycles per instruction) value is less than 1.5.	1.	In CISC, complex instructions consume multiple cycles to execute and an average CPI value lies between 2 and 15.
2.	It consists of less number of instructions.	2.	It consists of more number of instructions.
3.	It has fixed instruction format.	3.	It has variable instruction format.
4.	Hardware is responsible for executing instructions.	4.	Microprogram is responsible for executing instructions.
5.	It is highly pipelined.	5.	It is either less pipelined or not pipelined.
6.	The complexity exists in complier.	6.	The complexity exists in microprogram.
7.	It supports few addressing modes. Usually, instructions have register to register addressing mode.	7.	It supports many addressing modes.
8.	It posses multiple register set.	8.	It posses only single register set.

9.	Memory is referenced by less number of instructions.	9.	Memory is referenced by more number of instructions.
10.	The RISC architecture is as shown below,	10.	The CISC architecture is as shown below,

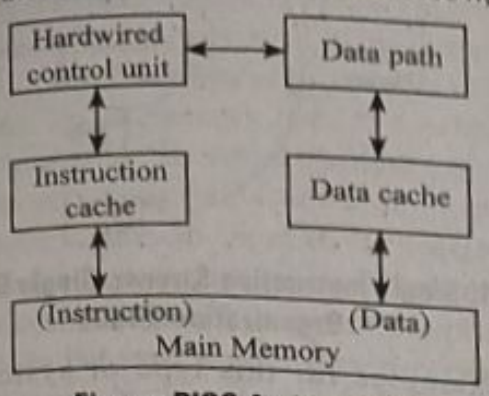


Figure: RISC Architecture

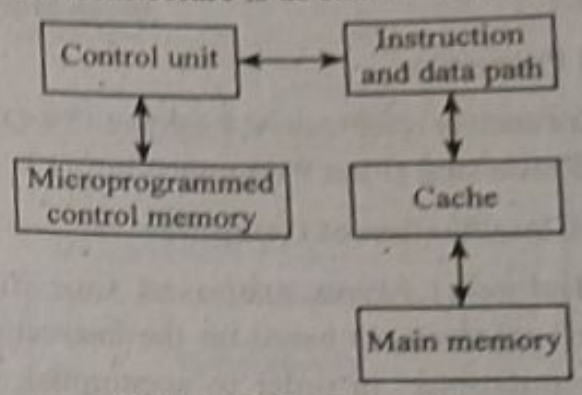


Figure: CISC Architecture

5.2 PIPELINE AND VECTOR PROCESSING

5.2.1 Parallel Processing

Q3. Discuss the concept of parallel processing.

Answer :

Parallel Processing

Parallel processing is a concept where concurrent execution of various tasks is accomplished. This is an important criteria for enhancing the overall throughput of the system. It is achieved by increasing the circuitry of a given system i.e., additional ALUs are used with high speed buses which are capable of carrying large bandwidth data. Parallel processing can be effectively accomplished by increasing the number of processors that are capable of operating parallely, i.e., if one processor is involved in executing a given computation, other processors can deal with various other computations thereby increasing the overall efficiency of the system.

Basically, two types of organizations are possible in parallel processing. They are,

(i) Distributed type (ii) Multiprocessor type.

(i) **Distributed Type:** Distributed type or distributed organization can be effectively utilized in solving a given computation where a group of computers (processing their own sets of memories and CPUs) situated at different zones work together.

(ii) **Multiprocessor Type:** Multiprocessor type or multiprocessor organization is a kind of organization where more than one processor acts parallely in order to execute a single or multiple independent tasks.

Consider the following organization where a single processor can be efficiently distributed among various processes.

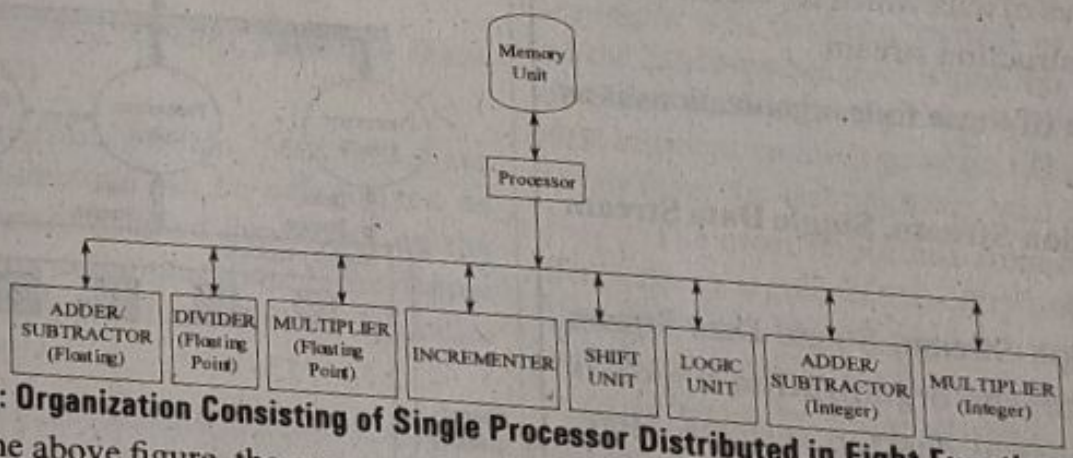


Figure: Organization Consisting of Single Processor Distributed in Eight Functional Units

As shown in the above figure, the processor has a direct contact with the main memory at one end and eight functional units on the other end. In order to execute several tasks, the processor distributes these tasks evenly among all these eight units depending on their designations. For example, assume that there are three tasks i.e., *A*, *B* and *C*. If task *A* represents an integer addition, then it is promoted to integer adder-subtractor unit from the given processor register. Similarly, if the other two tasks requires floating point division and multiplication, then they are promoted to floating point division and multiplication units respectively. Hence, these units work independently and accordingly produces the result, which is further transmitted to the respective destinations. Here, extremely sophisticated control units are required to handle these transactions. Moreover, even the processor should remain extremely capable enough to fulfill the requirements of these units.

Q4. What is parallel processing? Explain Flynn's classification of computer.

Answer : March-21(R18), Q7(a)

Parallel Processing

For answer refer Unit-V, Page No. 159, Q3, Topic: Parallel Processing (First Paragraph Only).

Flynn's Classification of Computer

Michael J.Flynn proposed four different computer organizations based on the instructions and data manipulations in order to accomplish parallel processing.

Types of Parallel Processors

These four organizations are as follows,

1. Single Instruction Stream, Single Data Stream (SISD)
2. Single Instruction Stream, Multiple Data Stream (SIMD)
3. Multiple Instruction Stream, Single Data Stream (MISD)
4. Multiple Instruction Stream, Multiple Data Stream (MIMD).

These four organizations focus on the way in which several instructions are executed on the available data resources. The two terms, data stream and instruction stream are frequently used. In general terms, the word stream is referred to as an array of entities. Hence, the word instruction stream specifies an array of large number of instructions, and the term data stream refers to those resources of data which are essential while executing the given instruction stream.

The illustration of these four organizations is as follows,

1. Single Instruction Stream, Single Data Stream (SISD)

Single Instruction Stream, Single Data Stream (SISD) is a uni-processor organization which is frequently found in normal desktop systems. Here, the instructions are executed consecutively, also this organization implements pipelined procedures. Hence, the instructions can also be executed in an overlapped fashion. As there is only a single control unit, all activities of the available functional units are governed by it.

The SISD organization can be diagrammatically represented as,

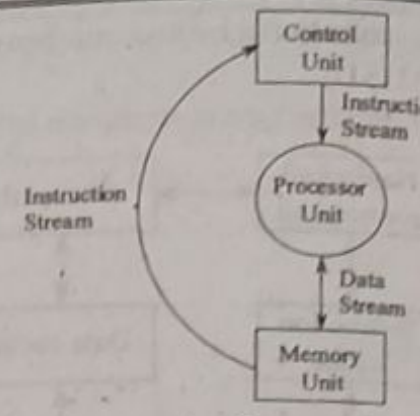


Figure-1: Single Instruction Stream, Single Data Stream Organization (SISD)

Examples for this type of systems include VAX–11/780, CDC 6600, IBM 370/168 and also the workstations that are similar to DEC, Hewlett Packard and Sun Microsystems.

2. Single Instruction Stream, Multiple Data Stream (SIMD)

Single Instruction Stream, Multiple Data Stream (SIMD) organization consists of a single control unit that governs an array of processors which are directly connected to multiple memory modules. These memory modules together form a single large memory unit. As the single memory unit is shared among several processors it is also referred to as shared memory unit. SIMD organization supports only a single control unit to govern all the activities of this system.

Hence, the software which is implemented in the control unit should be highly sophisticated. Even though, various processor units (in the given system) receive broadcasted messages from the single control unit, they work on different data streams.

The SIMD organization can be diagrammatically represented as,

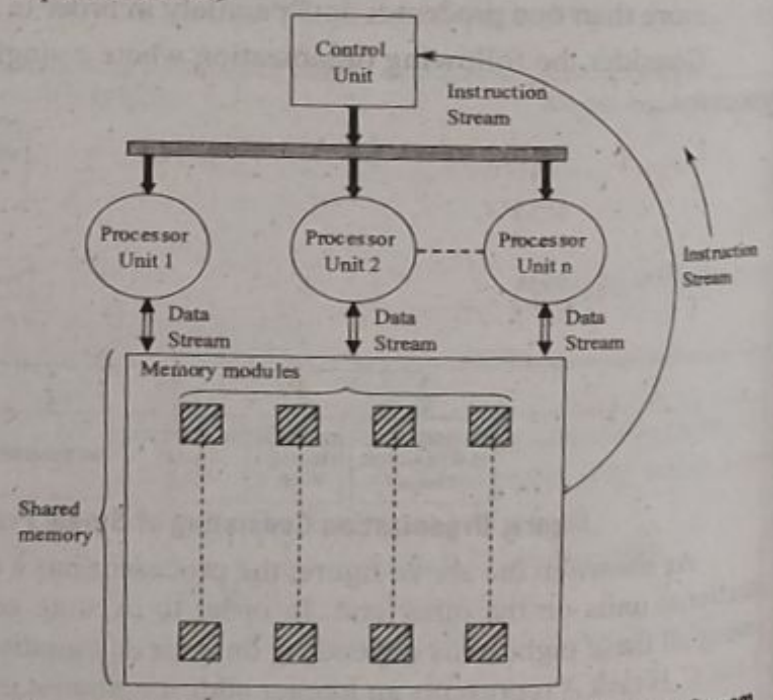


Figure-2: The Single Instruction Stream, Multiple Data Stream Organization (SIMD).

Examples of this type of systems include Thinking Machines CM-2, DAP, Illiac IV, MasPar MP-1 and MP-2, MPP and STARAN.

3. Multiple Instruction Stream, Single Data Stream (MISD)

Multiple Instruction Stream, Single Data Stream (MISD) architecture consists of a large number of control units with equal number of processors. As these processors share single memory unit, it is referred to as shared memory unit. The corresponding instruction streams which gets activated from their respective memory modules, forms an input to their associated control unit. The first processor receives data stream from the shared memory unit and the output of this processor forms an input to the next consecutive processor and the connection goes on. Finally, the last processor is connected to the memory. The MISD architecture is considered as an impractical architecture.

The MISD organization can be diagrammatically represented as,

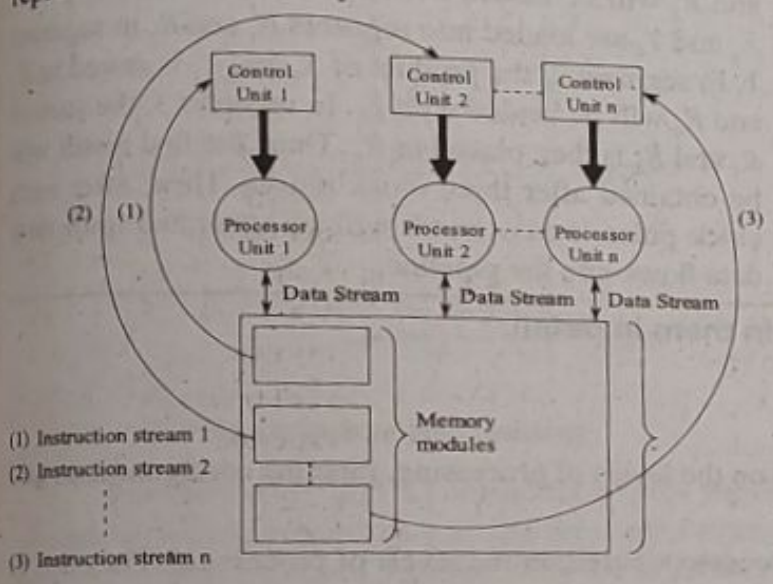


Figure-3: Multiple Instruction Stream, Single Data Stream Organization (MISD)

One of the example of MISD system is Beowulf design developed by Donald Becker at NASA.

4. Multiple Instruction Stream, Multiple Data Stream (MIMD)

Multiple Instruction Stream, Multiple Data Stream (MIMD) architecture can be categorized as tightly coupled or loosely coupled depending on the degree of inter-processor communication. If the degree of communication is high then the MIMD architecture is called tightly coupled.

When multiple instructions are performed on multiple data then the system is said to have MIMD architecture. Each processing element is assigned a separate control unit that receives the instructions from the shared memory and passes them to the respective Processing Elements (PEs). The PEs perform operations according to their instruction streams and stores the result back into the shared memory. The shared memory is divided into a number of modules in order to avoid read/write conflicts among PEs.

The MIMD architecture is shown below,

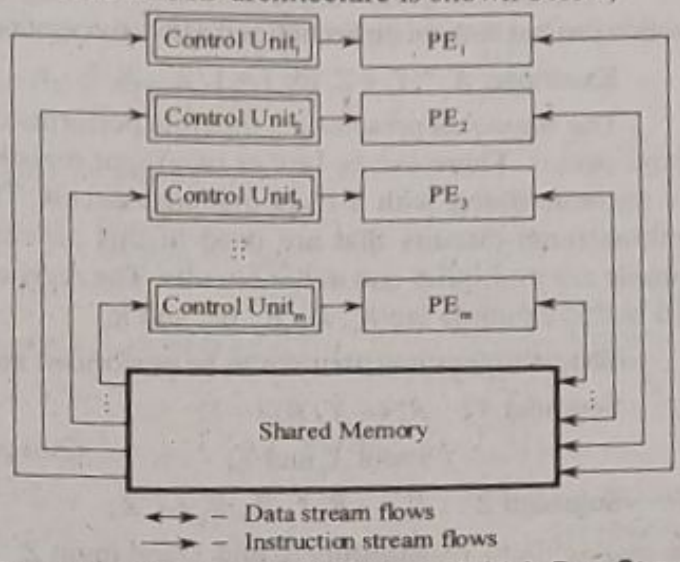


Figure-4: Multiple Instruction Stream, Multiple Data Stream (MIMD)

Some examples of such architecture are Denelcor HEP, C.mmp, Cray-2, S-1 and Burroughs D-825. However, when the degree of interaction is less then the resulting architecture is called loosely coupled. Tandem/16, univac 1100/80, IBM 370|168 MP, C.m*, IBM 3081/3084 are some examples of loosely coupled MIMD architecture.

5.2.2 Pipelining

Q5. What is pipelining? Explain pipeline processing with an example.

Answer : **Model Paper-I, Q10(b)**

Pipelining

A method that is used to decompose a sequential process into suboperations is called pipelining. In this pipelining method, all the subprocesses will be executed in a separate segment that can operate along with the remaining segments. Thus, when the binary information flows through several processing segments, it is said to be in pipeline. Each of these processing segments will perform partial processing depending on the partitioning performed on the task. The result obtained from one segment will be forwarded to the next segment and this goes on till the last segment. The result from the last segment will be the final.

The most important characteristic of pipeline is that, various segments can carry out their computations at the same time. Hence, separate registers are maintained with each segment. Because of this, it is possible to carry out operations on different data at the same time.

Structure of a Pipeline

A pipeline contains several segments. Each segment in turn contains an input register and a computational circuit. The input register contains the data, whereas, the computational circuit will perform the suboperation associated with that segment. The output of each combinational circuit forms the input for the next segment register. The registers are also connected to a clock, which allows the information flow across the pipeline at every clock pulse.

Consider the following example of solving the equation that has both multiplication and addition operations.

Example: $X_i * Y_i + Z_i$ for $i = 1, 2, ..., 8$

The segments present in a pipeline performs the suboperations. There can be one or two input registers in a segment along with a combinational circuit. The combinational circuits that are used in this pipeline example are multiplier and adder circuits. The registers used in this example are R_1, R_2, R_3, R_4, and R_5.

The suboperations that are to be performed are,

Segment 1 : $R_1 \leftarrow X_i$, $R_2 \leftarrow Y_i$;
Input X_i and Y_i

Segment 2 : $R_3 \leftarrow R_1 * R_2$, $R_4 \leftarrow Z_i$;
Multiply X_i and Y_i and input Z_i

Segment 3 : $R_5 \leftarrow R_3 + R_4$;

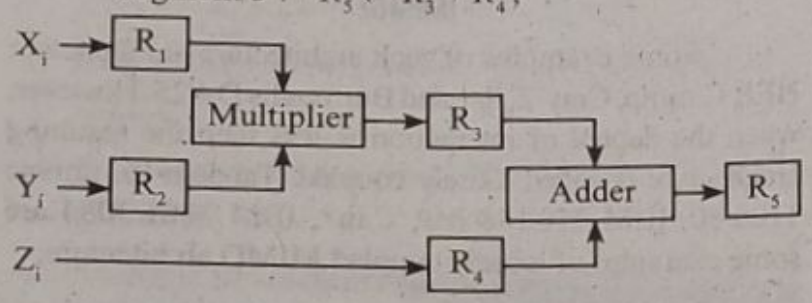


Figure: Pipeline Processing for the Given Example

After each clock pulse, all the five registers will be loaded with new data. This is shown in the following table.

Clock Pulse	Segment -1		Segment -2		Segment -3
	R_1	R_2	R_3	R_4	R_5
1	X_1	Y_1	$X_1 . Y_1$	Z_1	–
2	X_2	Y_2	$X_2 . Y_2$	Z_2	$X_1 * Y_1 + Z_1$
3	X_3	Y_3	$X_3 . Y_3$	Z_3	$X_2 * Y_2 + Z_2$
4	X_4	Y_4	$X_4 . Y_4$	Z_4	$X_3 * Y_3 + Z_3$
5	X_5	Y_5	$X_5 . Y_5$	Z_5	$X_4 * Y_4 + Z_4$
6	X_6	Y_6	$X_6 . Y_6$	Z_6	$X_5 * Y_5 + Z_5$
7	X_7	Y_7	$X_7 . Y_7$	Z_7	$X_6 * Y_6 + Z_6$
8	X_8	Y_8	$X_8 . Y_8$	Z_8	$X_7 * Y_7 + Z_7$
9	–	–	–	–	$X_8 * Y_8 + Z_8$
10					

In the first clock pulse, X_1 and Y_1 are loaded into registers R_1 and R_2. In the second clock pulse, the registers R_1 and R_2 are loaded with X_2 and Y_2 in segment 1. In segment 2, the product of X_1 and Y_1 is stored in R_3 and R_4 will be loaded with Z_1. In the third clock pulse, X_3 and Y_3 are loaded into registers R_1 and R_2 in segment 1. In segment 2, the product of X_2 and Y_2 is stored in R_3 and R_4 will be loaded with Z_2. In segment 3, the sum of R_3 and R_4 is then placed in R_5. Thus, the first result will be obtained after three clock pulses. Then, after each clock pulse, a new result will be generated until new data flows into the pipeline.

Q6. Classify the pipeline processors and explain them in detail.

Answer :

Classification of Pipeline Processors

The pipeline processors have been classified based on the levels of processing, pipeline configurations and control strategies.

Handler introduced the three classes of pipeline processors based on the levels of processing. They are as follows,

1. Arithmetic Pipelining

In this class of pipelining, the arithmetic logic unit of the processor is divided into number of segments to perform the operations of various data formats.

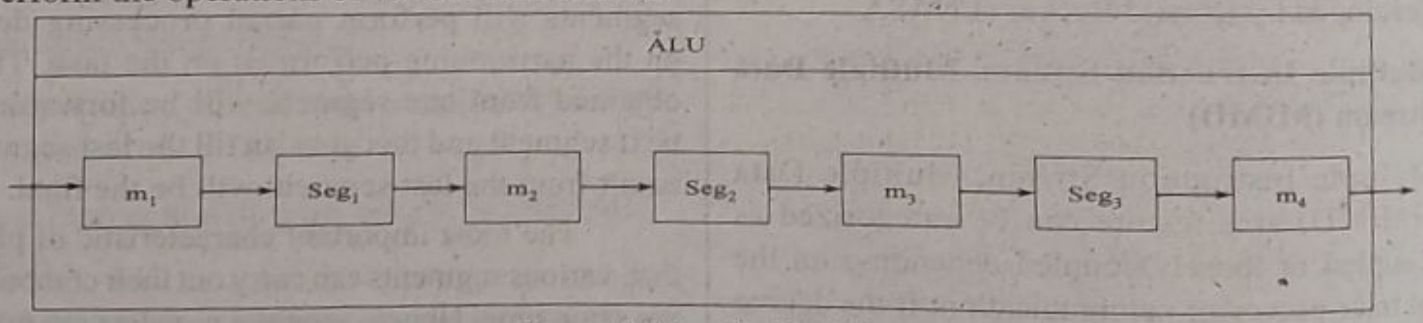


Figure: Arithmetic Pipelining

2. Processor Pipelining

In this class of pipelining, the processors are arranged in the cascaded form to perform the execution on the same data stream. These processors divide the task of a single data stream among themselves by passing the result of one processor to the subsequent processors through their respective memories.

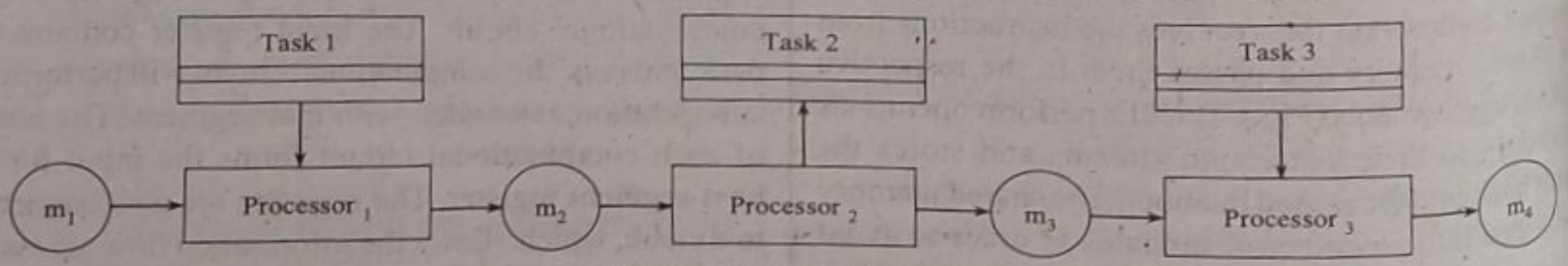


Figure: Processor Pipelining

Instruction Pipelining

3. In this class of pipelining, the execution of any instruction can be carried out simultaneously with the tasks of importing and decoding, and the tasks of improving the succeeding instructions, such process is called instruction look ahead process which is supported by most of the modern computers.

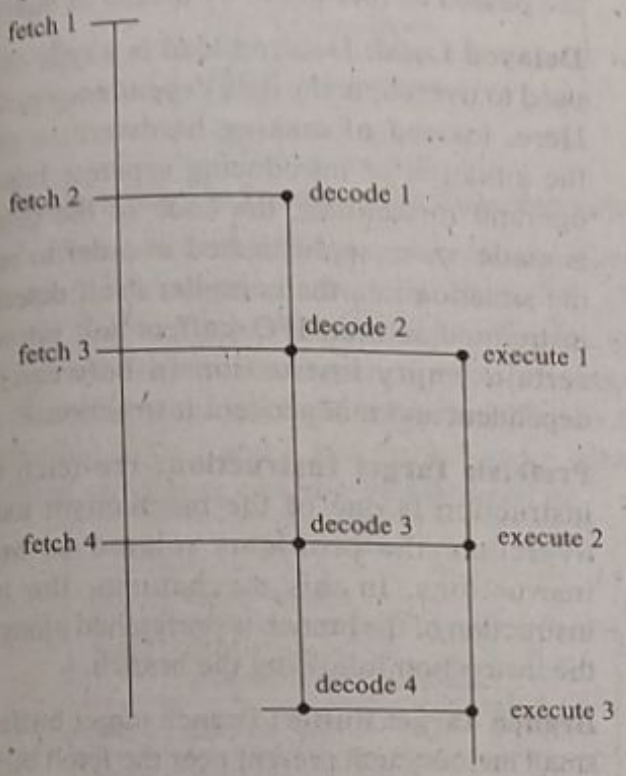


Figure: Instruction Pipelining

Ramamoorthy and Li introduced three other classes of pipeline processors based on the control strategies and configurations of pipeline. They are as follows,

(i) **Multifunction Pipelines:** In this class of pipeline processor, the pipeline joins different subsets of stages in the single pipeline in order to achieve the execution of multiple functions at the same time or at different times.

(ii) **Dynamic Pipelines:** The dynamic pipeline is multifunction and provides the facility of the installation of more than one functional configurations.

(iii) **Vector Pipelines:** The vector pipeline processors are controlled by specialized firmware and hardware that allows to operate on vector instructions using vector operands.

5.2.3 Arithmetic Pipeline, Instruction Pipeline

Q7. Explain pipeline for floating-point addition and subtraction.

Answer :

Consider the following diagrammatic representation of the sequence of steps involved in performing floating-point addition and subtraction.

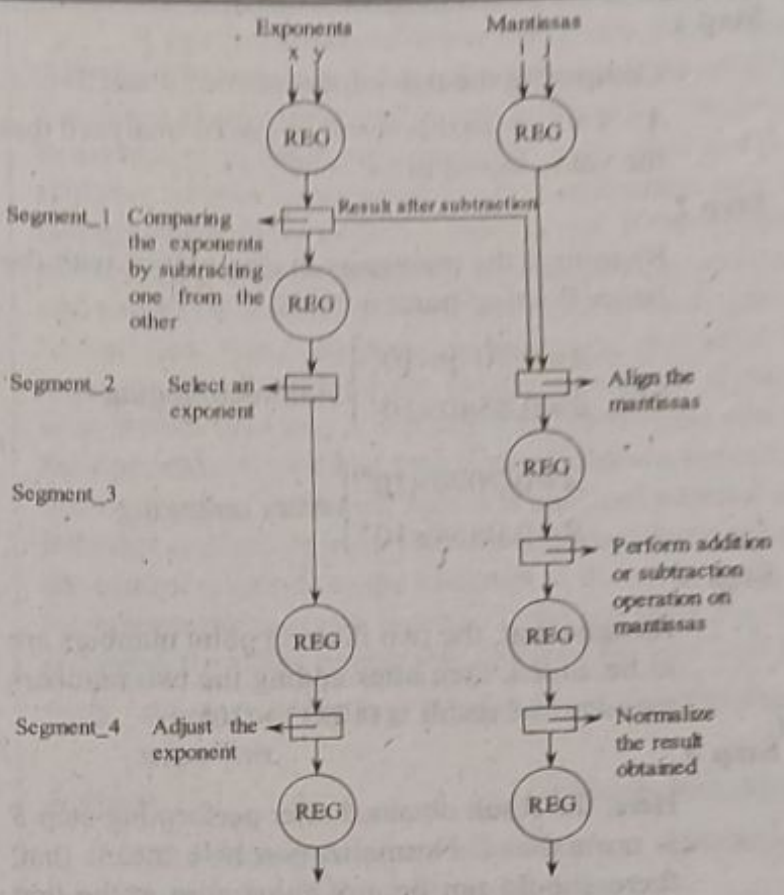


Figure: Arithmetic Pipeline

The steps involved while performing the pipelining of floating-point addition and subtraction are as follows,

Step 1: Comparisón of Exponents

Initially, the two given exponents are compared and the result is obtained. This result is used to determine the larger as well as the smaller number.

Step 2: Alignment of Mantissas

Then, the mantissas are rearranged in accordance with the larger floating-point number.

Step 3: Addition/Subtraction of Mantissas

Next, if addition is to be performed, then both the mantissas are added or if subtraction is to be performed, then the mantissa of the smaller floating-point number is subtracted from the mantissa of greater floating-point number.

Step 4: Normalizing Result

Finally, the result obtained after performing step 3 is normalized in accordance with the rules of computer arithmetic.

Example

In order to understand the above logic, consider the following example,

Consider two floating point numbers of the form,

$$A = x2^i, \quad B = y2^j$$

Let the two numbers be,

$$A = 0.7079 \times 10^4$$

$$B = 0.8540 \times 10^3$$

Step 1

Comparing the two exponents i.e., 4 and 3

$4-3=1$; with this result, it can be analyzed that the value stored in 'A' > 'B'.

Step 2

Rearrange the mantissas in accordance with the larger floating-point number.

i.e., $\left.\begin{aligned} A&=0.7079\times10^4\\ B&=0.8540\times10^3\end{aligned}\right\}$ Before arranging

$\left.\begin{aligned} A&=0.7079\times10^4\\ B&=0.0854\times10^4\end{aligned}\right\}$ After arranging

Step 3

Assume that, the two floating point numbers are to be added, then after adding the two numbers the required result is 0.7933×10^4

Step 4

Here, the result obtained after performing step 3 is normalized. Normalization here means that, there should not be any value zero at the first digit position of the fractional part of the required answer. As the answer obtained does not have a value zero at the first digit position of the fractional part, hence it does not require any further normalization.

Q8. Explain the following terms with related to the instruction pipeline,

(a) Data dependency
(b) Hardware interlocks
(c) Operand forwarding
(d) Delayed load
(e) Prefetch target instruction
(f) Branch target buffer.

Answer :

(a) **Data Dependency:** In pipeline processing, assume that the two instructions say I_1 and I_2 are being executed. If the processing of instruction I_2 depends on the processed result of instruction I_1 then this consequence is referred to as data dependency.

(b) **Hardware Interlocks:** Hardware interlocks is a method used to overcome the problems caused by data dependency i.e., the instruction contains operands on which the other subsequent instructions (which are stationed in a FIFO buffers) depends on. In order to resolve this problem, hardware interlocks detect these instructions in their respective FIFOs and make the execution of current instruction to be delayed for several clock cycles. Since, the delaying of the current instruction execution is caused due to this specific hardware interlock method.

(c) **Operand Forwarding:** Operand forwarding is also one of the method to resolve the data dependency conflicts. Here, the instructions which depend on the instruction currently under execution are detected and their desired dependent values are supplied to them by means of special buses arranged to satisfy the current purpose. As the dependent values (i.e., operands) are passed or forwarded by means of buses.

(d) **Delayed Load:** Delayed load is a type of logic used to overcome the data dependency problem. Here, instead of making hardware to control the situation or introducing separate buses for operand forwarding, the code of the compiler is made more sophisticated in order to resolve the situation i.e., the compiler itself detects the instruction in the FIFO buffers and introduces certain empty instruction in between these dependent and independent instruction.

(e) **Prefetch Target Instruction:** Pre-fetch target instruction is one of the mechanism used to overcome the problems related to branch instructions. In this mechanism, the target instruction of the branch is prefetched along with the instruction following the branch.

(f) **Branch Target Buffer:** Branch target buffer is a small memory unit present near the fetch block of the instruction execution. It consists of two fields, the first field indicates the address of the previously executed branch instruction and the next field indicates the target instruction of the branch. The branch target buffer also stores few instruction after storing the branch target instruction.

Q9. What is meant by instruction pipeline? Explain four segment instruction pipeline.

OR

Explain about instruction pipelining with an example. Aug./Sep.-22(R18), Q8(a)

OR

Explain four segment instruction pipeline.

(Refer Only Topic: Four Segment Instruction Pipeline

Answer : April/May-23(R18), Q10(b)

Instruction Pipeline

An instruction pipeline is a sequence of independent steps with storage at the termination of each step. The instruction pipeline speeds the instruction execution by overlapping the execution of the current instruction with the fetch, decode and operand inorder to fetch subsequent instructions. This technique is also referred to as 'instruction look ahead'.

Example: Four Segment Instruction Pipeline

The complete execution of a given instruction involves six steps. These steps are tabulated below,

Steps	Description
Step 1	Instruction fetch
Step 2	Instruction Decode
Step 3	Effective address calculation
Step 4	Operands fetch
Step 5	Instruction execution
Step 6	Store the result

Table-1: Steps in Executing the Given Instruction

The above table gives a general idea behind the execution of a simple instruction. But, CPU supports many of such instructions which does not traverse through all the above mentioned steps. For simplicity, the above six steps are reduced to four. This is done by combining steps 2 and 3 and steps 5 and 6. Hence, the resultant sequences can now be tabulated as,

Steps	Description	Symbol
Step 1	Instruction fetch	F
Step 2	Instruction decode + Effective address calculation	IE
Step 3	Operands fetch	O
Step 4	Instruction execution + Store the result	IS

Table-2: The Six Instructions Reduced to Four

Now, consider the flowchart depicting the four segment CPU pipeline.

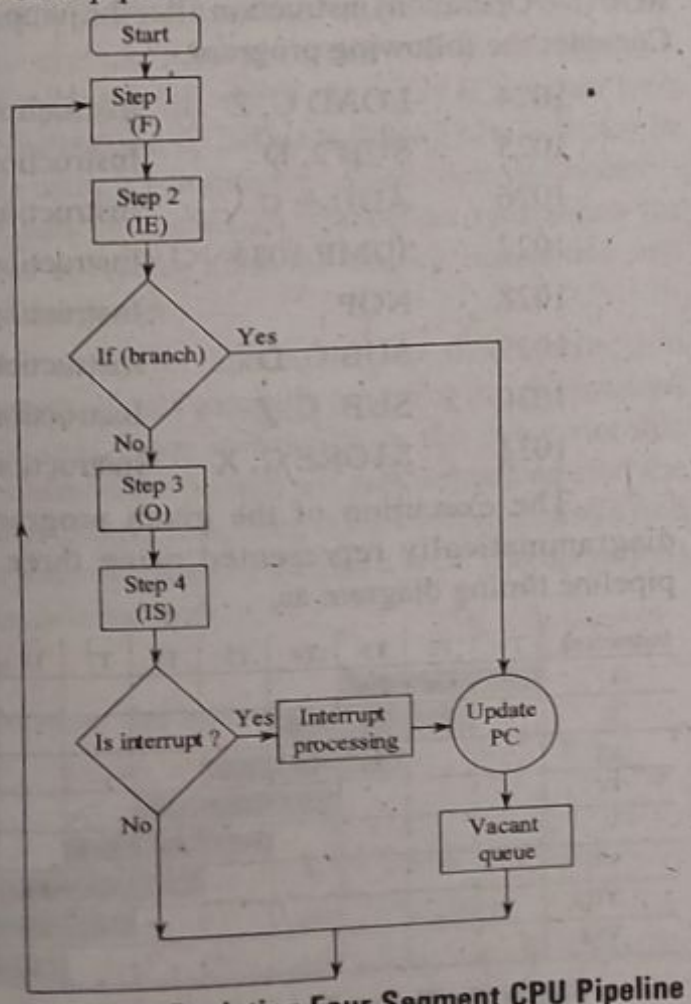


Figure: Flowchart Depicting Four Segment CPU Pipeline

It can be observed from the given flow chart that, the CPU execution begins with execution of step 1, where a given instruction is fetched from the memory. In the next step, the given instruction is decoded and the effective address is evaluated. Further, in the next step, it is checked whether a branch is made or not. If there exists a branch, then the program counter register is updated and the FIFO buffers (which are used to store other instructions, when the current instruction execution is in progress) are made vacant. On the other hand, if there is no branch then step 3 and step 4 are performed where the operands are fetched and instructions are executed thereby storing the result, then it is checked whether an interrupt is made or not. If an interrupt is detected then the control responds to the interrupt at the same time, it increments the program counter. If the interrupt has not occurred, then the control gets diverted back to step 1.

Q10. Give the timing diagram of instruction pipeline.

Answer : **Model Paper-I, Q11(a)**

The timing diagram of a 4-stage instruction pipeline is shown in the following figure,

Step / Instruction	1	2	3	4	5	6	7	8	9	10	11
I1	F	IE	O	IS							
I2 (Branch)		F	IE	O	IS						
I3			F	–	–	F	IE	O	IS		
I4				–	–	–	F	IE	O	IS	
I5								F	IE	O	IS

Figure: Timing Diagram of a 4-stage Instruction Pipeline

In the 4-stage instruction pipeline, if branch instructions are not included, then each of the four segments *F*, *IE*, *O* and *IS* will operate on different instructions. Where, *F* represents instruction fetch, *IE* represents instruction decode with effective address calculation, *O* represents operands fetch and *IS* represents instruction execution with result storing.

For example, consider step 3 in the above timing figure. In this step, the operand required in executing instruction '*I*1' is fetched, instruction '*I*2' is decoded, and instruction '*I*3' is fetched in segments *O*, *IE* and *F*, respectively. Because, the branch instruction '*I*2' was not yet decoded.

If a branch instruction is included, then as soon as it is decoded, all the other instructions which are to be decoded are halted until the branch instruction allows to branch to a new instruction, then this new instruction will be fetched. And, if the branch instruction does not allow to branch, then the instruction, which is already fetched in the previous step is decoded.

For example, consider step 3 in the above timing diagram. In step 3, the branch instruction, i.e., instruction '*I*2' is decoded. But, as soon as it completes decoding, it stops other instructions from being decoded in the preceding steps until it finishes its execution.

The instruction '*I2*' takes two more steps to finish its execution i.e., step 4 and step 5: As the branch instruction may or may not allow to branch to a new instruction, the respective instructions will be either fetched or decoded, accordingly.

If the branch instruction allows to branch to a new instruction, then new instruction will be fetched in step 6. And, if the branch instruction i.e., '*I2*' does not branch to a new instruction, then the next instruction i.e., '*I3*' which was fetched in step 3 will be decoded in step 6. In step 7, the next instruction i.e., '*I4*' will be fetched. This process continues, until the next branch instruction appears in the pipeline.

Q11. Discuss the various conflicts that might arise in a pipeline. How are they resolved?

Answer :

Dec.-19(R18), Q10(a)

Various Conflicts Arise in Pipeline

Pipeline conflicts arises when the pipeline processing deviates from its normal execution due to certain conditions/causes. The causes for pipeline conflicts are as follows,

1. **Branch Difficulties:** Certain times while implementing pipelining it may happen that few instructions or branches may effect the program counter, causing its value to be incremented or decremented in an uncertain manner. This would either lead to a problem called "branch difficulties".
2. **Data Dependency:** Sometimes it may happen that, the value of an operand belonging to an instruction depends on the execution of other instruction. During such conditions erroneous results may occur. Such a problem is referred to as data dependency problem.
3. **Resource Conflicts:** During pipelining, simultaneous execution of two instructions causes conflict in a given resource. Such a problem is referred to as "resource conflicts". This can be avoided by using separate memories for instruction and data.

Resolutions of Pipeline

(i) RISC

For answer refer Unit-V, Page No. 158, Q1 Topic: RISC, (First Paragraph Only).

(ii) Delayed Branch

For answer refer Unit-V, Page No. 167, Q13, Topic: Delayed Branch.

5.2.4 RISC Pipeline

Q12. Discuss about RISC Pipeline.

March-21(R18), Q8(a)

OR

Explain three segment instruction pipeline. Show the timing diagram with data conflict.

Answer :

The three segment pipeline timing in case of RISC processor can be accomplished by considering NOP (No Operation) instruction before the jump or branch instruction. Assume that, here each instruction has to jump through three stages referred to as instruction fetch (F) decode (De) and execute (Exe) stages, respectively.

Consider the following program where no NOP instruction is required.

1024	LOAD C, Z	Instruction I
1025	SUB 2, D	Instruction II
1026	ADD 5, C	Instruction III
1027	JUMP 1030	Instruction IV
1028	SUB C, D	Instruction V
1029	SUB C, Z	Instruction VI
1030	STORE C, X	Instruction VII

The execution of the given instructions can be diagrammatically represented as,

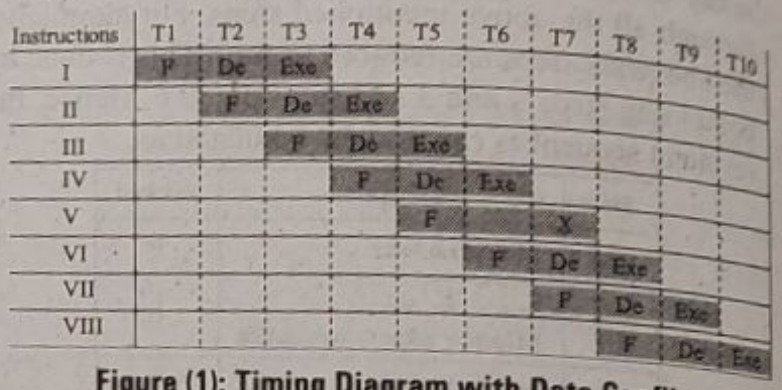


Figure (1): Timing Diagram with Data Conflict

The 'X' mark at time period 'T7' in figure (1) is due to the jump made at instruction 5. As a result, instruction 8 gets executed, due to which the data belonging to instruction 6 has to be flushed out. Hence, this problem can be overcomed by the introduction of NOP (No Operation) instruction after the jump statement. Consider the following program,

1024	LOAD C, Z	Instruction I
1025	SUB 2, D	Instruction II
1026	ADD 5, C	Instruction III
1027	JUMP 1031	Instruction IV
1028	NOP	Instruction V
1029	SUB C, D	Instruction VI
1030	SUB C, Z	Instruction VII
1031	STORE C, X	Instruction VIII

The execution of the given program can be diagrammatically represented using three segment pipeline timing diagram as,

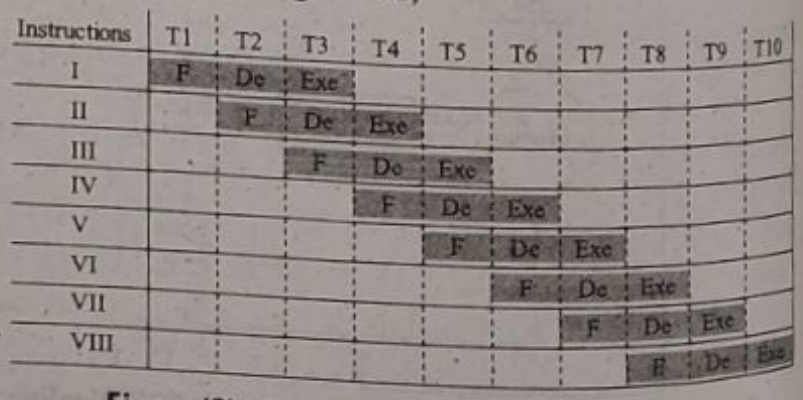


Figure (2): Timing Diagram with Delayed Load

As seen from figure (2), the introduction of NOP instruction does not cause any effect on pipelining instead it makes the execution to proceed smoothly.

Advantages of RISC Over CISC

The various advantages of RISC over CISC are as follows,

(i) The instructions supported by RISC are simple and can be executed within a single clock cycle.

(ii) The instructions formats are fixed i.e., they are predefined.

(iii) Only few instructions are sufficient for solving a given problem.

(iv) RISC supports only fewer addressing modes, most of them supports register to register transfers.

(v) The instructions supported by RISC exerts high degree of pipelining.

(vi) RISC supports multiple register sets.

(vii) Delays in RISCs are avoided to a larger extent since only a few memory transfer instructions are supported by RISC.

Q13. What do you mean by delayed branch? Explain with an example.

Answer :

Delayed Branch

A branch instruction is an instruction that delays the execution of other pipeline operations. The delay is due to the need for updating the Program Counter (PC) and fetching instruction at the specified branch location. Until the instruction at the specified branch address is not fetched, no other operation such as fetch, decode, or execute can be carried out in the pipeline. The clock cycle during which the delay is resulted from a branch instruction is called a "delay slot". It can be avoided by using a mechanism called "delayed branch". The compilers in the RISC processors rearranges the instructions before or after the branch instruction and fills in the delay slot.

The compiler can also perform instruction rearrangement. It first carefully analyzes the instructions and inserts only those instruction in the delay slot that neither influences the branch nor gets influenced after the execution of branch. However, if the compiler cannot find such instructions, it simply pushes a NOP (no operation) instruction into the slot.

Example

Consider the following code,

Move	x	to	R0
Decrement	R0	by	1
Multiply	R5	and	R6
Branch	to	address	A

Figure (1) depicts the status of the pipeline without using the delayed branch mechanism.

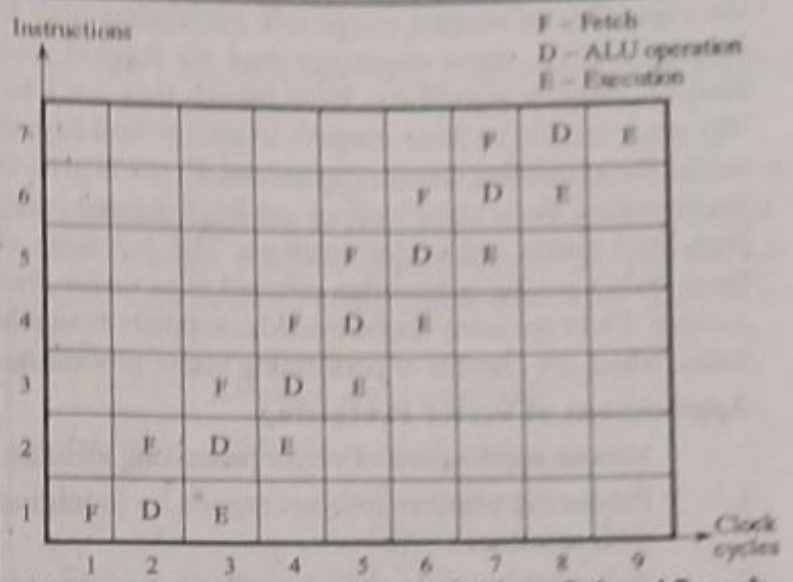


Figure (1): Status of the Pipeline Without Delayed Branch

First instruction corresponds to move, second to decrement, third to multiply, fourth to branch, fifth and sixth to NOP (PC is updated) and seventh to the instruction which specifies branch to address A. Thus, during clock cycles, 5 operations in the pipe are halted and resumed during clock cycle 6, after PC is updated.

When delayed branch mechanism is used the pipe status looks like,

Figure (2): Status of Pipeline after using Delayed Branch

Instead of decrementing and multiplying before the branch instruction, the compiler moves them into the delay slot which are executed after updating PC. Thus, the rearranged code becomes,

Move	x	to	R0
Branch	to	address	A
Decrement	R0	by	1
Multiply	R5	and	R6

5.2.5 Vector Processing, Array Processor

Q14. Write short notes on vector processing.

Model Paper-I, Q11(b)

OR

Illustrate vector operations and vector processing.

Answer : March-21(R18), Q7(b)

Vector Processing

There are many science and engineering fields that demand extremely complex manipulations, which are beyond the reach of normal computers. This is because,

the capability of normal computers can extend only to certain level i.e., these computers does not support such kind of complex algorithms. Even though they are forcibly implemented on these systems, it takes around days to weeks for completing these computations. Hence in general mathematics, these computations are implemented using high-level matrix and vector notations. The execution of these vector notations are often referred to as *vector processing*. There are many sophisticated computers available today, which are capable of performing vector processing.

Applications of Vector Processing

Various applications of vector processing includes,

1. Producing weather forecast reports by indulging with satellites.
2. Temperature and other quantity measurements performed during satellite launching.
3. In scanning image processing which is useful in determining one's identity.
4. Finding petroleum ores by examining the samples.
5. Earthquake measurement equipment.
6. Sophisticated calculations involved during human genome mapping.
7. Medical diagnosis.
8. Many artificial intelligence enabled systems like robots, etc.

Operations of Vector Processing

When there are large arrays of numbers on which arithmetic operations are to be performed, they can be expressed as vectors and matrices of floating point numbers. An ordered set of data items arranged in an array is referred to as a 'vector'. It can be a one-dimensional or a two-dimensional array. A vector can be represented as follows,

(i) **Row Vector:** If the data item in a vector 'V' are arranged in a row wise manner, it is called a row vector.

Representation : $V = [V_1, V_2, ..., V_n]$ where n is the length of the vector.

(ii) **Column Vector:** If the data items in a vector V are arranged in a column wise manner, it is called a column vector.

Representation : $V = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ V_n \end{bmatrix}$ where n is the length of the vector.

A general purpose computer carries out the computation by iterating through each element of the array. Thus, the vector operations must be divided into several individual computations with each of them having a distinct subscripted variable. For example, if 'V' is a vector then the element V_x is represented as $V(X)$. Here 'X' corresponds to the address (memory or register) where the corresponding number is located.

Example

Consider a vector subtraction operation, $\boxed{R = P - Q}$ where, P, Q and R are vector of length 10. In this operation, vector R is obtained by subtracting Q from P. A FORTRAN do loop that carries out this vector subtraction operation is as follows,

$$DO\ 12\ X = 1,10$$
$$12R(X) = P(X) - Q(X)$$

When a vector processor is employed, it does not need to fetch and execute all the 10 instructions, as a vector operation can be specified in a single instruction as given below,

$$R(1:10) = P(1:10) - Q(1:10)$$

Q15. Explain how matrix multiplication can be performed using vector processors.

Answer :

In a matrix multiplication, the number of inner products or multiply-add operations involved depend on the size of the matrix.

Two $n \times n$ matrix multiplication requires n^2 inner products or n^3 multiply-add operations,

For example, if P and Q are two 3×3 matrices such that,

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \text{ and } Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}$$

then,

$$P \times Q = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \times \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Where,

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \text{ is the resulting matrix.}$$

In order to calculate the resulting matrix R, the inner product is calculated as follows,

$$\boxed{r_{ij} = \sum_{k=1}^{3} p_{ik} \times q_{kj}}$$

It requires $n^2 = 3^2 = 9$ inner products or $n^3 = 3^3 = 27$ multiply-add operations.

The inner product can be calculated on a pipeline vector processor. An inner product consisting of k-product terms is given as,

$$\boxed{R = P_1Q_1 + P_2Q_2 + P_3Q_3 + ... + P_{k-1}Q_{k-1} + P_kQ_k}$$

Calculating Inner Product on a Pipeline Vector Processor

A pipeline to calculate inner product consists of a floating point multiplier pipeline and a floating point adder pipeline. Suppose, each of these pipelines have four segments. Initially, these segment registers will hold zeroes.

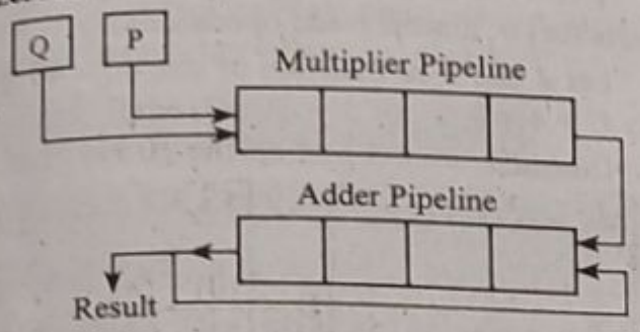


Figure: Pipeline Vector Processor to Calculate Inner Product

The elements of P and Q matrices may reside in a memory or a processor register. These values are multiplied in pairs as $(P_i \times Q_i)$, one in every cycle. After the first four cycles, the four multiplier segments will contain products of pairs from P_1Q_1 through P_4Q_4. In the fifth cycle, P_1Q_1 will be the output to the adder pipeline from multiplier pipeline, and a '0' will be added to it. Similarly, in the 6th, 7th and 8th cycle, P_2Q_2, P_3Q_3 and P_4Q_4 will be the respective outputs to the adder pipeline and they will be added with zeroes. While P_1Q_1 to P_4Q_4 pairs are output to adder pipeline, the pairs P_5Q_5 to P_8Q_8 are multiplied and are placed in the four multiplier segments. After first eight cycles, the four adder segments will contain P_1Q_1 through P_4Q_4 pairs and the four multiplier segments will contain P_5Q_5 to P_8Q_8 pairs. In the eighth cycle, when P_8Q_8 is placed in the multiplier pipeline, at the same time, P_5Q_5 will be the output of multiplier pipeline and P_1Q_1 will be the output of adder pipeline. In the ninth cycle, P_5Q_5 and P_1Q_1 will be added in the adder pipeline. Similarly, in the 10th, 11th and 12th cycles, $A_2B_2 + A_6B_6$, $A_3B_3 + A_7B_7$ and $A_4B_4 + A_8B_8$ are calculated, respectively. In the 13th cycle, $A_1B_1 + A_5B_5$ will be added to A_9B_9. When all the products in multiplier pipeline are output to the adder pipeline, zeroes will be inserted into it, by the system. And, the four adder segments will contain the following partial products,

$$R = \underbrace{P_1Q_1 + P_5Q_5 + P_9Q_9 + P_{13}Q_{13} + P_{17}Q_{17} + \ldots}_{\text{In 1st segment}}$$

$$\underbrace{+ P_2Q_2 + P_6Q_6 + P_{10}Q_{10} + P_{14}Q_{14} + P_{18}Q_{18} + \ldots}_{\text{In 2nd segment}}$$

$$\underbrace{+ P_3Q_3 + P_7Q_7 + P_{11}Q_{11} + P_{15}Q_{15} + P_{19}Q_{19} + \ldots}_{\text{In 3rd segment}}$$

$$\underbrace{+ P_4Q_4 + P_8Q_8 + P_{12}Q_{12} + P_{16}Q_{16} + P_{20}Q_{20} + \ldots}_{\text{In 4th segment}}$$

These four partial sums are then added to obtain the required final sum.

Q16. Explain the array processors.

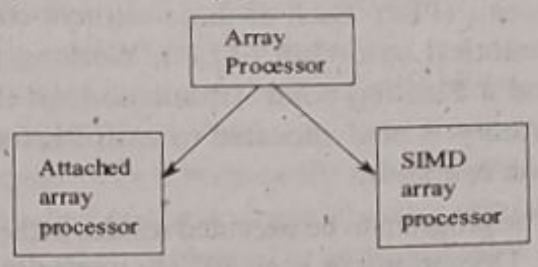
Answer : (Model Paper-II, Q10(a) | April/May-23(R18), Q11(b))

Array Processor

An array processor refers to a special category of processors which was developed to perform computations on larger volumes (array) of data. The two categories of organizations which come under array processors are,

(a) Attached array processor

(b) SIMD array processor.



(a) Attached Array Processor

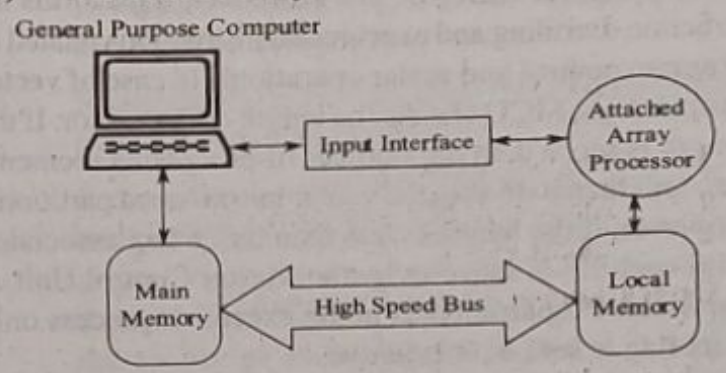


Figure: Attached Array Processor Organization

An attached array processor organization enhances the capabilities of local computer to perform complex vector manipulations. In order to perform this task, it introduces several arithmetic and logic units which are well-versed to handle complex floating-point addition and multiplication computations at the same time implementing pipelined features. The attached array processor interacts with the normal computer by means of an external I/O controller. The local memory has a direct contact with the attached array processor. Moreover, the local memory also forms a connection with the main memory via high speed bus. When the external array processor is not considered, the computer behaves like a normal general-purpose computer. As the attached array processor interacts with this computer by means of an I/O interface, hence the computer treats this processor as an external I/O device (the name of the processor).

(b) Single Instruction Multiple Data (SIMD) Array Processor

A computer with multiple processing units, capable of performing a single instruction on multiple data streams is called SIMD (Single Instruction Multiple Data) array processor The organization of an SIMD array processor is shown in the figure,

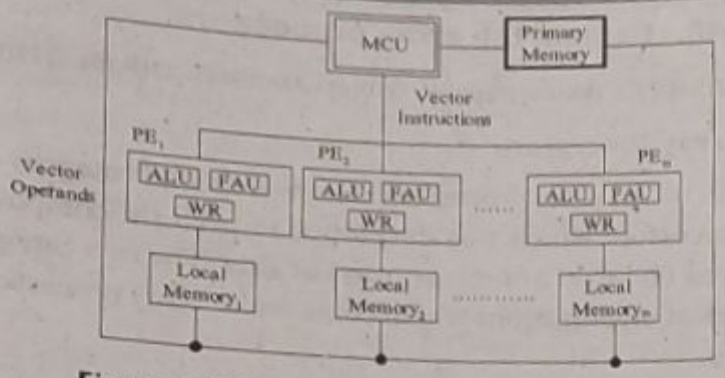


Figure: SIMD Array Processor Organization

The SIMD array processor includes 'm' Processing Elements (PEs). Each of these element consists of an Arithmetic Logic Unit (ALU), Working Register (WR) and a Floating-point Arithmetic Unit (FAU). A local memory is also allocated to each PE for storing instruction operands.

The program to be executed resides in the primary memory. Operations in each PE are controlled by the Master Control Unit (MCU). Moreover, it performs instruction decoding and executes all instructions related to program control and scalar operations. In case of vector instructions, MCU checks the length of the vector. If the length is more than the number of processing elements i.e., 'm', then it divides the vector into m-word partitions. However, if the length is less than 'm', a flag associated with each PE is set/reset by the Master Control Unit. A particular PE participates in the execution process only if its flag is set i.e., it is active.

Once the vector length is known and m-word partitions or selected PEs are set, then the master control unit broadcasts vector operands to the local memory of participating PE's. Finally, the vector instructions are broadcasted and executed parallely.

Example

For the vector multiplication $S = E * F$, the k^{th} ($k = 1, 2, ..., m$) components e_k and f_k are first stored in each PE's local memory. Then the instruction $s_k = e_k * f_k$ is broadcasted to all PE's. Each PE stores the s_k component in their local memories. Thus, many instructions are executed in one cycle.

ILLIAC IV computer is the most popular SIMD array processor regardless of its parallel processing efficiency. It is no more in use, SIMD processors can only be applied to vector or matrix computations but not to any other type of computation. Moreover, they cannot be used in the context of data-processing programs.

Q17. Consider the multiplication of two 40 × 40 matrices using a vector processor.

(a) How many product terms are there in each inner product and how many inner products must be evaluated?

(b) How many multiply add operations are needed to calculate the product matrix?

Answer : Dec.-19(R18), Q11

Multiplication of Two 40 × 40 Matrices using a Vector Processor

In a matrix multiplication, the number of inner products (or) multiply-add operations involved depend on the size of the matrix,

Two $n \times n$ matrix multiplication requires n^2 inner products (or) n^3 multiply-add operations.

Let A and B be matrices of size 40×40 and the matrix $C = A \times B$.

Consider

$$A = \begin{bmatrix} a_{11} & \cdots & a_{140} \\ \vdots & & \vdots \\ a_{401} & \cdots & a_{4040} \end{bmatrix} \quad B = \begin{bmatrix} b_{11} & \cdots & b_{140} \\ \vdots & & \vdots \\ b_{401} & \cdots & b_{4040} \end{bmatrix}$$

$$C = A \times B$$

$$C = \begin{bmatrix} a_{11} & \cdots & a_{140} \\ \vdots & & \vdots \\ a_{401} & \cdots & a_{4040} \end{bmatrix} \times \begin{bmatrix} b_{11} & \cdots & b_{140} \\ \vdots & & \vdots \\ b_{401} & \cdots & b_{4040} \end{bmatrix}$$

$$C = \begin{bmatrix} c_{11} & \cdots & c_{140} \\ \vdots & & \vdots \\ c_{401} & \cdots & c_{4040} \end{bmatrix}$$ is the resulting matrix

The number of product term in each inner product is $n = 40$ because if the inner product,

$$C_{11} = a_{11}b_{11} + a_{12}b_{21} + a_{13}b_{31} \ldots\ldots + a_{140}b_{401}$$

Then, it has number of product term equal to n. Therefore, the number of inner product can be evaluated as,

$$n^2 = 40 \times 40 = 1600$$

$$\therefore \quad n^2 = 1600$$

Since, the inner products are $C_{11}, C_{12}, C_{13}, \ldots\ldots C_{4040}$ and they are $n \times n = n^2$.

(a) Calculating Inner Product on a Pipeline Vector Processor

For answer refer Unit-V, Page No. 169, Q15, Topic: Calculating Inner Product on a Pipeline Vector Processor (First Paragraph Only).

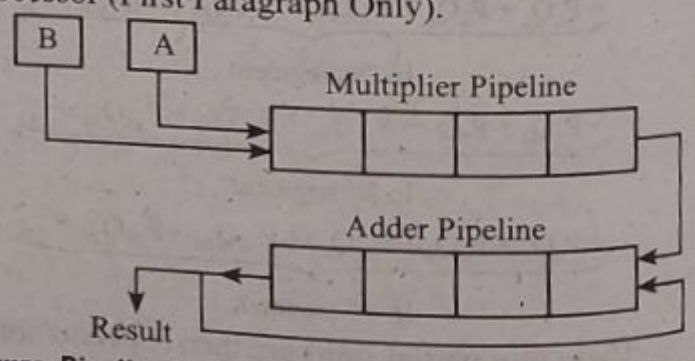


Figure: Pipeline Vector Processor to Calculate Inner Product

There are 40 Product terms in each inner product, i.e., $n = 40$.

$$n^2 = n \times n$$

$$40^2 = 40 \times 40 = 1600$$

Therefore, 1600 inner product must be evaluated, one for each element of the product matrix.

(b) The number of multiple add operations are needed to calculate the product matrix is, $n^3 = (40)^3 = 64{,}000$.

5.3 MULTIPROCESSORS

5.3.1 Characteristics of Multiprocessors

Q18. Discuss the characteristics of multiprocessors.

Answer : **(Model Paper-II, Q10(b) | March-22(R18), Q8(b))**

A system supporting interconnection structure of more than one processor is usually referred to as multiprocessor system. In this system, there are multiple independent processors which can execute several concurrent tasks. The main objectives of multiprocessor system is to enhance the overall efficiency and performance of the system.

Following are the important characteristics of multi-processors.

1. **Occupies Very Less Space and also they are of Low Cost**

This is one of the major characteristic of multiprocessor system which has added to its fame in the recent years. Though a single system supports the existence of more than one processor, it is widely used due to its low cost and size of the processor. This is made possible due to the emergence of the most sophisticated technology referred to as very large scale integrated chip technology or in short VLSI which facilitated the integration of several millions of transistors into a single minute chip at a very low cost.

2. **Enhances the Reliability of the Existing System to a Great Extent**

At certain times, it may happen that either hardware or software faults may corrupt the operation of a given processor. During these conditions, the disability of one processor will have little or no effect on the operation of other processors since, they all function independently. This also gives rise to fault tolerance capability of a system, as the left over work of the corrupted processor is assigned to other processors. In this way, the corruption of one processor will not halt the execution process of the entire system, rather it decreases efficiency. All the tasks are executed and thus, increasing the reliability of the system.

3. **Enhances the Overall Performance of the System and hence Increasing the Throughput**

The main objective of the multiprocessor system is to enhance the overall performance of the system. In this regard, the given multiprocessor system can be effectively utilized in two ways i.e., either by dividing a single large task into multiple task blocks and assigning each of them to a separate processor (or) assigning these processors with multiple independent tasks.

By analyzing first consideration, following conclusions can be made.

- When a single large task block is divided into multiple task blocks and are assigned to individual processors in an independent manner, system's performance can be enhanced to a large extent. This is because the time required to solve this task using a single processor is extremely high when compared to multiprocessor system.
- Dividing and assigning of tasks to the processors can be done in two ways,

(i) It can be done before dividing these tasks to their respective executable blocks.

(ii) It can also be done by making the compiler compatible ehough to detect the parallelism in a given program and assigning each of its tasks to their respective processors. This also enhances the ability of the compiler to detect the data dependency instructions i.e., the instructions which depends on other instructions for their execution and hence executing them prior to other instructions. This makes the process of resolving conflicts between them easily.

By analyzing the statement "Multiple independent tasks can be assigned to each of these processors", following conclusions can be made.

- Each processor can guard different functionalities of a single organization.
- If a banking sector is considered, a single processor can be assigned to update day-to-day transactions of the customers, while other processors can be given the responsibility of maintaining the database of all employees. In this regard, the processor maintains a given set of ALUs that can govern the customers transactions and the processors which frequently interacts with database (according to queries supplied to them) can be assigned to handle basic employee's details.

Q19. Explain briefly about tightly coupled and loosely coupled multiprocessors.

Answer :

Multiprocessors are classified into two different types based on their memory organization i.e.,

1. **Tightly-coupled Multiprocessors:** This type of multiprocessor is also called a "shared memory multiprocessor" as this system employs a common shared memory. This shared memory is accessible by all the processors connected to it. That means, the data stored in this memory can be shared by all these processors. The processors in this system also have their own local memories called as 'cache memory'. A tightly-coupled multiprocessor system is generally employed when the degree of interactions between the tasks is high.
2. **Loosely-coupled Multiprocessors:** This type of multiprocessor is also called a "distributed memory multiprocessor". In this multiprocessor system, processors are associated with their own private local memories. A switching scheme is used to put together all these processors. This scheme routes information between these processors with the help of a message-passing scheme. With this scheme, the processors will be able to transfer data and programs to other processors in the form of packets. Each packet holds an address, data and an error detection code. This type of communication system being used will transfer a packet either to the nearest available processor or to any given processor. A loosely-coupled multiprocessor is very efficient when the degree of interactions between the tasks is very low.

5.3.2 Interconnection Structures

Q20. Explain with the help of a neat sketch how a time shared bus interconnection system for multiple processors provides a common communication path connecting all of the functional units.

Answer :

Time-shared Common Bus Organization

The time-shared common bus organization includes a single common bus shared by multiple processors. This organization can be diagrammatically represented as shown in the following figure.

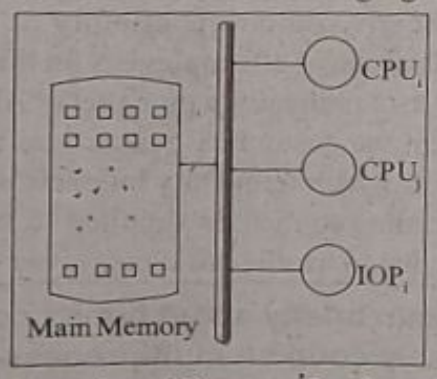


Figure: Time-shared Common Bus Organization

From the above figure it can be observed that, a time shared common bus organization consists of a shared common bus among multiple processors. There is a single memory unit (referred as main memory) which is also connected to the common bus. Whenever any of these processors intends to communicate either with the memory or with other processors, it (processor) has to initially check the availability of the given bus. If the bus is busy, then the processor either waits till the bus is available or switches to other tasks and tries later. If the bus is available, the given processor issues a command (related to any of the destinations say memory or other processors etc) and releases it on the bus. The command is checked by all the attached devices and response is made only by the corresponding device whose address matches with the command address. In this way, the transaction gets initiated. On the completion of the transaction, the bus is released.

It can be analyzed from the above mentioned illustration that, while one device is using the bus, all the other devices should either switch on to some other process or remain idle. Also, only one processor is authorized to use the bus at any given instant of time. Conflicts arise when two or more devices seek the bus at the same time. One way to overcome these conflicts is to the introduce separate unit referred to as bus controller. It checks to see that, the conflict is resolved either by providing priorities to each of these devices or by assigning time slots to them so that they will get equal chance of sharing the bus.

However, the other measures can introduce multiple bus systems. But, this increases cost as well as the complexity of the system. Following is an economical time shared common bus organization which maintains the dual bus structure.

System Bus Structure for Multiprocessors

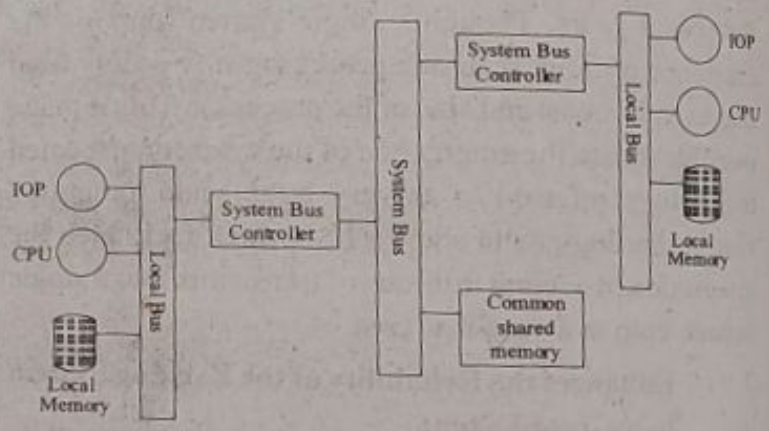


Figure: System Bus Structure for Multiprocessors

In this structure, a system bus is connected to one or more system bus controllers and each system bus controller is connected to a local bus. A local bus is in turn connected to one or more processors such as CPU, IOP and a local memory. A local processor can access local memory as well as the I/O devices which are connected to local IOP. The system bus is also connected to a common shared memory. This memory is accessible by all the processors connected to the system bus. But, at a given time only one processor can access the shared memory. While one of the processors is busy in accessing the common shared memory, other

processors will communicate with their local memory and I/O devices. If the system bus is directly connected to an IOP, then all the I/O devices connected to that IOP can be accessed by all the processors. If a part of local memory is allowed to work as a cache memory connected to CPU, then the cycle time of this CPU can be allowed to reach the average access time of local memory.

Q21. Explain multiport memory organization with a neat sketch.

Answer :

Multiport Memory Organization

It is a system which has multiple memory ports with each of these processors connected to all other memory units by means of separate buses. Following figure shows three CPUs and three memory units connected through a number of buses.

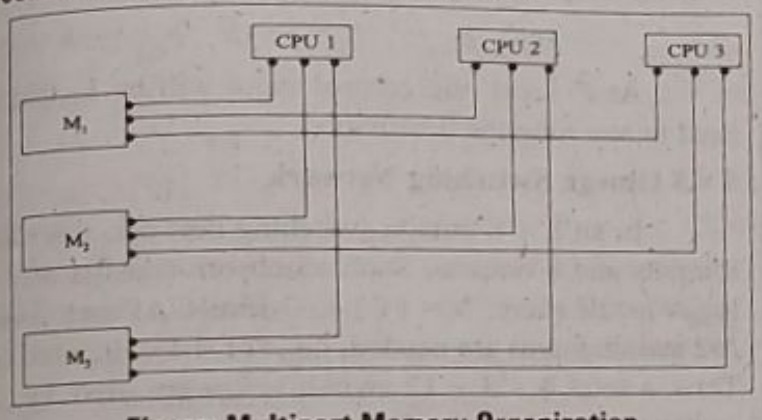


Figure: Multiport Memory Organization

The bus which originates from CPU is a combination of address, data and control lines, respectively. It connects to the respective memory units through their ports. Hence in the above interconnection structure, each memory unit maintains three ports, with each port accommodating single processor bus. This memory unit must have an internal control logic so as to determine which processor can access the memory. Since, all the CPUs are connected to all the memory modules, there exist certain conflicts. To avoid these conflicts, each CPU is assigned with some priority based on its port positions. For example, the CPU1 will have the highest priority and CPU3 will have the lowest priority.

Advantage

The major advantage of using this architecture is high data transfer rate, as each processor is maintaining separate bus lines to connect to these memory units.

Disadvantage

The disadvantage of this system is that it cannot be implemented for the systems containing a large number of processors. As the number of processors increases, the complexity of the system also increases which inturn increases the cost of the overall system.

Q22. What are the different kinds of multistage switching networks? Explain with neat sketch and compare their functioning.

OR

Construct a diagram for a 4 × 4 omega switching network. Show the switch setting required to connect input 3 to output.

(Refer Only Topics: Omega Switching Network, 4 × 4 Omega Switching Network)

Answer : **(Model Paper-II, Q11(a) | Sep.-21(R18), Q5(a))**

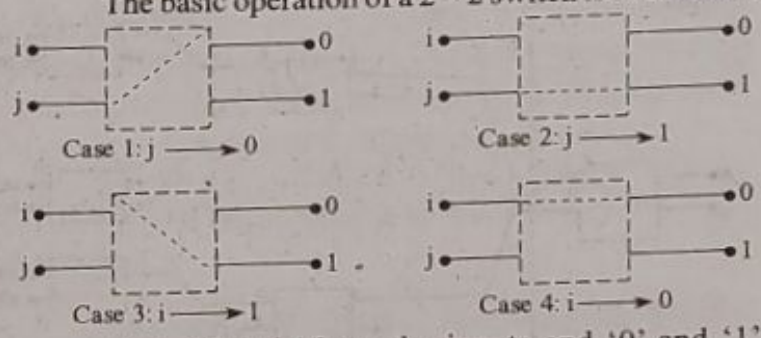
Multistage Switching Network

A network that supports multistage switches for activating the transmission between the pair of nodes (processor and memory unit) is often referred to as *multistage switching network*. This type of network allows existence of multiple parallel paths between a given set of nodes.

A multistage network is less expensive than cross-bar switches. A general multistage network has *m* inputs and *m* outputs. The most basic structure of a multistage network consists of two inputs and two outputs. This structure is used as a building block for building a very large network.

2 × 2 Switches

The basic operation of a 2 × 2 switch is as follows,



Here, '*i*' and '*j*' are the inputs and '0' and '1' are the outputs. A switch can route any of these inputs to the requested output. If the two inputs '*i*' and '*j*' request to two different outputs simultaneously then both the requests can be processed, whereas if both the inputs request the same output then one of the requests is blocked till the first request is processed. This interconnection between inputs and outputs is possible due to the presence of control logic in these switches.

Inorder to interconnect 2^m modules, a network with *m* stages is used. In such a case, there always exists a path between any two modules of the network. But, there are certain conditions where multiple requests are not satisfied parallely. The following figure shows a multistage network.

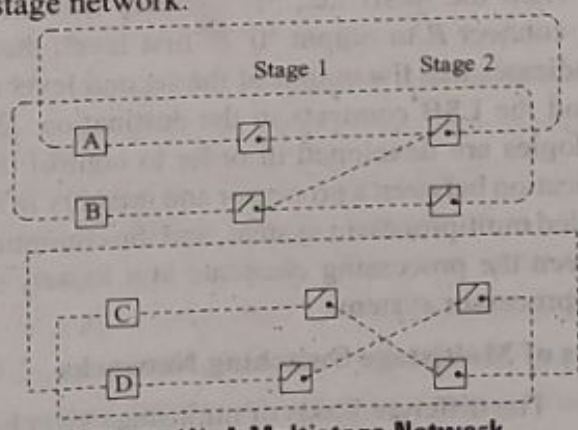


Figure (1): A Multistage Network

In a multistage interconnection structure consisting of 'm' nodes, the expression representing the total number of switches can be given as,

$$4 \times \frac{m}{2} \times \log_2 m \Rightarrow \boxed{2m \log_2 m}$$

Where $\log_2 m$ corresponds to the number of stages holding $m/2$ switch boxes at each stage. Note that, each switch box can accommodate a total of four switches. The above expression specifies that, the multistage interconnection network can be applied to large networks since the configuration of switches to be used will be less when compared to other networks, such as crossbar switches, where the total number of switches to be used is m^2. The routing method employed in a multistage network is quite simple and easy. Inorder to understand the routing procedure, consider a simple binary tree with 2 × 2 switches as shown in the below figure.

Binary Tree Network with 2 × 2 Switches

Let A and B be the two different modules connected to eight different memory units through switches.

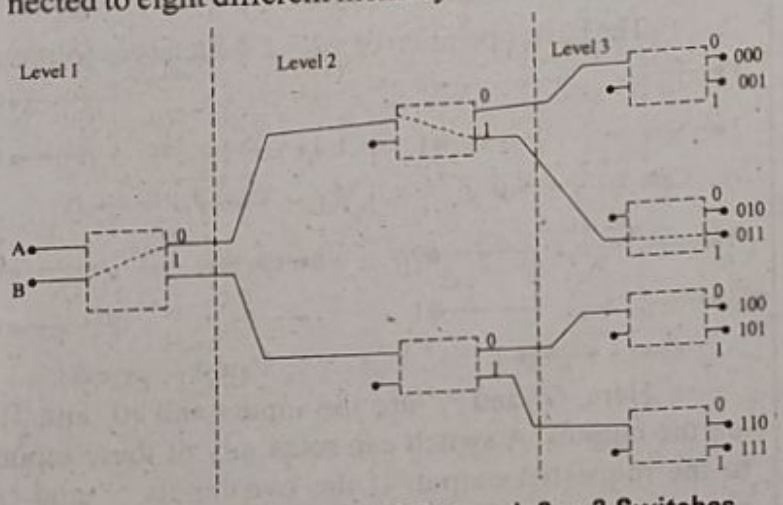


Figure (2): Binary Tree with Network 2 × 2 Switches

It can be observed from the above figure that, there are certain binary bits which are located at the output of the figure. These bits play a vital role in determining the path in the above circuit. As the above figure maintains three bits at each output, the figure is also divided into three levels. Each bit corresponding to each output exerts its significance at different levels (i.e.,) first or most significant bit is considered in level 1, the second bit in level 2 and so on. Consider an example where processor B is connected to the memory module 011. Then the MSB i.e., '0' signifies that the switch must connect B to output '0' at first level, the second bit indicates that the output of the second level must be '1' and the LSB connects to the destination. Different topologies are developed in order to control the communication between a processor and memory in a tightly coupled multiprocessor system, and the communication between the processing elements in a loosely coupled multiprocessor system.

Types of Multistage Switching Networks

The different kinds of multistage switching networks are as follows,

1. Omega Switching Network

Omega switching network is also called a shuffle exchange network because it uses shuffle and exchange functions. The inputs are shuffled using a shuffle function which is given as,

$$S(b_{x-1}\, b_{x-2}\, b_{x-3} \ldots\ldots b_0) = (b_{x-2}\, b_{x-3} \ldots\ldots b_0\, b_{x-1})$$

Where,

$(b_{x-1}\, b_{x-2}\, b_{x-3} \ldots\ldots b_0)$ = Source address

$(b_{x-2}\, b_{x-3} \ldots\ldots b_0\, b_{x-1})$ = Destination address

and x = Number of levels.

The exchange function is employed within the switch. It connects the input to upper output of the switch if the control bit is 0, and it connects to the lower output if the control bit is 1, the exchange function, E of a switch controlled by input bit, b_0 is given as,

$$E(b_{x-1}\, b_{x-2} \ldots\ldots b_1\, b_0) = (b_{x-1}\, b_{x-2} \ldots\ldots b_1\, \bar{b}_0)$$

At i^{th} level, the control input will be b_i that is used in reaching the destination.

8 × 8 Omega Switching Network

In an 8 × 8 omega switching network, there are 8 inputs and 8 outputs. Such a network consists of $x = \log_2 N$ levels (here, $N = 8$), i.e., 3 levels. At each level, $N/2$ switch boxes are needed, i.e., $8/2 = 4$ switch boxes. Thus, a total $4 \times 3 = 12$ switch boxes are used, i.e., 4 switches in each of the three levels.

The connections in an 8 × 8 omega switching network are shown in below figure.

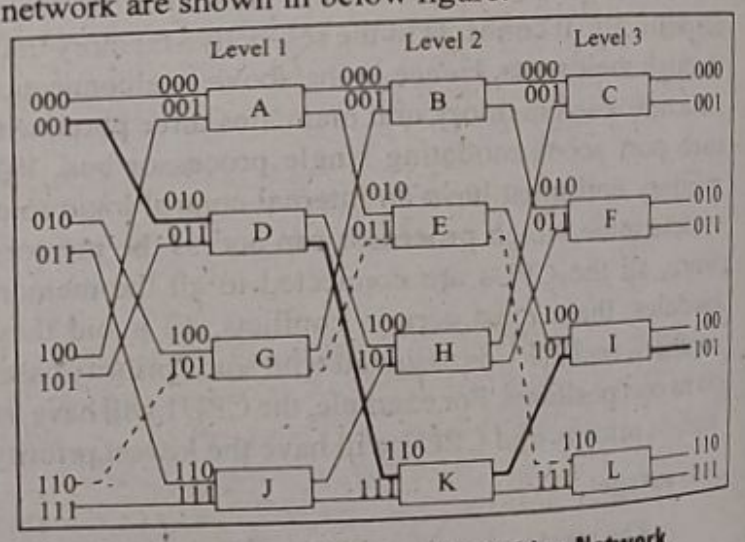


Figure (3): 8 × 8 Omega Switching Network

All the switch boxes from A to L in the above switching network are 2 × 2 crossbar switches that can route any of the two inputs to any of the two outputs. In this topology, there exists a single path from a given source to a destination.

The interconnections between switches are made using the shuffle function,

$$S(b_2\, b_1\, b_0) = (b_1\, b_0\, b_2)\ [\because x = 3] \quad \ldots (1)$$

Example

If input $b_2\, b_1\, b_0$ is 110 then its output from the switch at level 1 is connected to $b_1\, b_0\, b_2$ i.e., 101 using equation (1). Similarly, other connections are also made.

As, $S(b_2\ b_1\ b_0) = (b_1\ b_0\ b_2)$

At level 1, $S(1\ 1\ 0) = (1\ 0\ 1)$

At level 2, $S(1\ 0\ 1) = (0\ 1\ 1)$

At level 3, $S(0\ 1\ 1) = (1\ 1\ 0)$

These interconnections are shown in the figure (3) as dashed lines.

A source initiates a request by forwarding a 3-bit pattern associated with a destination. This 3-bit pattern is moved along the network through three levels i.e., level 1, level 2 and level 3. The most significant bit among the 3-bits is identified in the level 1, and its corresponding switch will route it to the upper or lower output if that bit is 0 or 1, respectively. Similarly, level 2 identifies middle bit and level 3 identifies least significant bit and accordingly their switch boxes will route to the bit of their upper or lower output.

Consider an example that source 001 needs to communicate with destination 101. The source sends the destination address 101. As the MSB is 1 (∵ $\boxed{1}01$), at level 1, switch routes to the lower output. The middle bit is '0' (∵ $1\boxed{0}1$) at level 2, switch routes to the upper output. Similarly, as the LSB is 1 (∵ $10\boxed{1}$), at level 3, switch routes to lower output, and reaches the destination (101). This path from the source to the destination is shown in figure (3) as a dark line.

4 × 4 Omega Switching Network

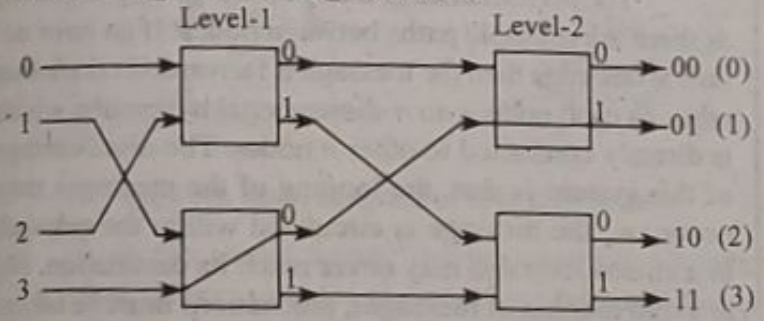


Figure (4): 4 × 4 Omega Switching Network

2. Butterfly Switching Network

This network uses butterfly switches. If N is the number of inputs and outputs in a network, then total number of levels 'x' required for the input source to reach an output destination can be determined using the formula $x = \log_2 N$ with each level having $N/2$ switches. Thus, for 8 inputs and outputs, a network with 3 levels will be built (as, $\log_2 8 = 3$), and each level will have $\frac{N}{2} = \frac{8}{2} = 4$ switches.

At any stage 'p', the butterfly function is given as,

$$F_p(b_{x-1}\ b_{x-2} \ldots\ldots b_{p+1}\ b_p\ b_{p-1} \ldots\ldots b_1\ b_0) = (b_{x-1}\ b_{x-2} \ldots\ldots b_{p+1}\ b_0\ b_{p-1} \ldots\ldots b_1\ b_p)$$

Where,

$(b_{x-1}\ b_{x-2} \ldots\ldots b_{p+1}\ b_p\ b_{p-1} \ldots\ldots b_1\ b_0)$ = Source address

$(b_{x-1}\ b_{x-2} \ldots\ldots b_{p+1}\ b_0\ b_{p-1} \ldots\ldots b_1\ b_p)$ = Destination address

p = Any value from 1 to $x - 1$

The connections in a butterfly network are shown in the following figure.

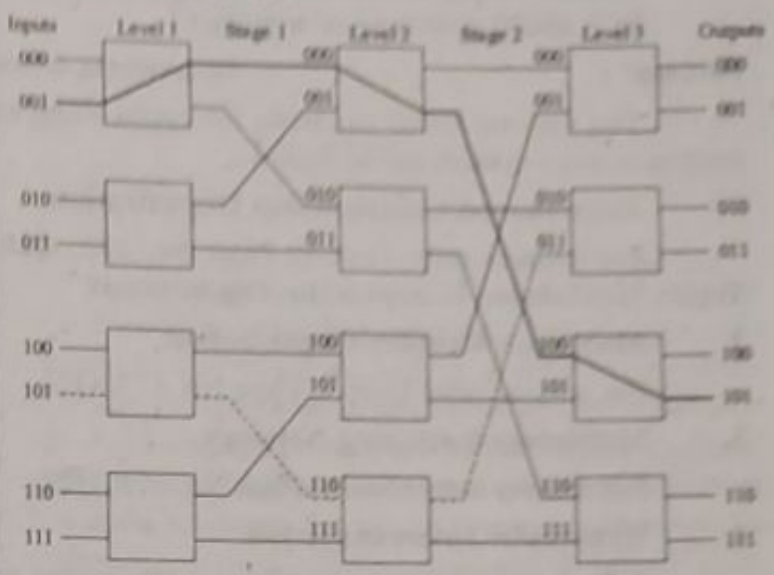


Figure (5): A Butterfly Network with Three Stages

This network contains 8 inputs and 8 outputs with 3-levels, where each level contains $\frac{N}{2} = \frac{8}{2} = 4$ switches.

Thus, a total of 12 switches are employed. The interconnections between switches at p^{th} stage (where, $p = 1, 2$) are made using the following butterfly function.

$$F_p(b_2\ b_1\ b_0) = (b_2\ b_0\ b_1), \text{ when } p = 1$$

$$F_p(b_2\ b_1\ b_0) = (b_0\ b_1\ b_2), \text{ when } p = 2$$

Example

If the input $b_2\ b_1\ b_0$ is 101, then its output from the switch at stage $p = 1$ is connected to $(b_2\ b_0\ b_1)$ i.e., (110). At stage $p = 2$, it will be connected to $(b_0\ b_1\ b_2)$, i.e., 011. This interconnection is shown in figure (5) as a dashed line.

In this network, a source initiates a request by forwarding a 3-bit pattern associated with the destination.

The routing within a switch is performed by considering the $(l+1)^{th}$ MSB bit of destination address in a level 'l'. A '0' will make the data to be routed to upper output, whereas a '1' will make the data to be routed to lower output.

Consider an example, that source 001 needs to communicate with destination 101. At level $l = 0$, the $(l+1)^{th}$ bit i.e., $b_{l+1} = b_{0+1} = b_1$ of the destination address considered. As $b_1 = 0$ (∵ $1\boxed{0}1$), the data is routed to the upper output of the switch from source 001. At level $l = 1$, the $(l+1)^{th}$ bit (i.e., b_2) is considered. As $b_2 = 1$ (∵ $\boxed{1}01$), the data is routed to the lower output of the switch. Similarly, at level $l = 2$, b_3 is considered. As $b_3 = 1$ (∵ $10\boxed{1}$), the data is routed to the lower output and the destination 101 is reached. This communication is shown in figure (5) as a dark line.

Q23. What are the various forms available for establishing an interconnection network in a multi processor system?

Answer : Dec.-18(R16), Q11(b)

The various forms available for establishing an multiprocessor system are as follows,

1. Time-shared Common Bus Organization

For answer refer Unit-V, Page No. 172, Q20, Topic: Time-shared Common Bus Organization.

2. Multiport Memory Organization

For answer refer Unit-V, Page No. 173, Q21.

3. Multistage Switching Network

For answer refer Unit-V, Page No. 173, Q22.

4. Hypercube Interconnection

A hypercube interconnection system is used to implement a network which consist of 2^n nodes in an n-dimensional cube. This structure reflects loosely coupled multiprocessor system with each node correspond to a processor. Moreover, the processor may associate certain additional circuitry such as I/O interfaces and can also maintain local memories etc.

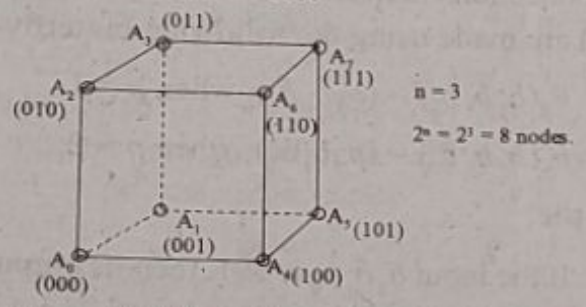


Figure (1): Three-dimensional Hypercube

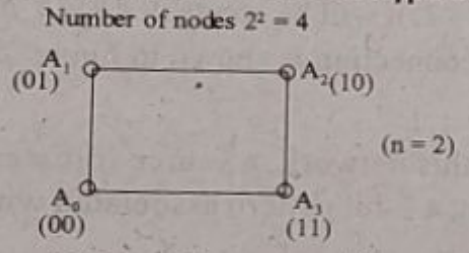


Figure (2): Two-dimensional Hypercube

Nodes A_0, A_1, A_2,... A_n in the above figures represent the different processors. All the processors are connected through a communication path (referred to as edges). In this structure each node is assigned a unique binary address such that, if the interconnection structure consists of 2^n nodes, then each node can have 'n' bit binary address (Eg. If $n = 2$, then this organization consists of $2^2 = 4$ nodes with each node maintaining a 2 bit address). Moreover, a given node in the above mentioned interconnection structure differs from all it's neighbouring nodes by 1 bit in terms of their addresses.

Now, for every multiprocessor system one of the major issue to be considered is routing of data from a given source node to the destination node. For a given node, if it intends to transmit or route the data to its direct neighbouring nodes (the nodes which differ in one bit position from the source node) it requires to route the data through a single edge to reach them (if shortest path is considered).

Similarly, for a given node (source node), in order to route the data to a destination node, (whose binary address differs by 2 bits with respect to source node), the source needs to route the data through two edges in order to reach the corresponding destination.

For example, in the three-dimensional hypercube structure, if the node say, A_0 with binary address (000), intends to transmit data to node say A_2 with binary address (010), then it requires to route the data only through a single edge. Since, the source node address and the destination node address differ only by a single bit position. Similarly, if A_0 intends to transmit data to node A_7, then it needs to route the data through three edges since both source node (A_0) address and the destination node address (A_7) differs by 3 bit positions.

When multiple interconnection structure is considered i.e., $n > 3$, the routing issues may then turn out to be more complex. In order to reduce this complexity to a certain extent, a simple routing scheme is implemented. It suggests that, in order to determine the number of edges through which the data is to be routed from a given source to a given destination, both of these addresses (source and destination) should be XORed and the resulting 1's will be the number of edges through which the data has to be routed i.e., If the result of XORing of two addresses is four 1's, then the data is required to pass through four edges to reach the destination.

The performance of the system is greatly improved as there are multiple paths between nodes. If an error occurs at one edge then the message is forwarded via another edge, as each node is an n-dimensional hypercube which is directly connected to other n nodes. The disadvantage of this system is that, the looping of the messages may occur i.e., the message is circulated within the network in a closed loop and may never reach its destination. So, to avoid looping of messages, precautions must be taken.

Intel iPSC and NCUBE/ten are the examples of machines which have used hypercube interconnection networks.

5. Crossbar Switch Network

The crossbar switch network structure contain several cross points that provide communication links between several processors and memory units. The following figure shows a crossbar switch network.

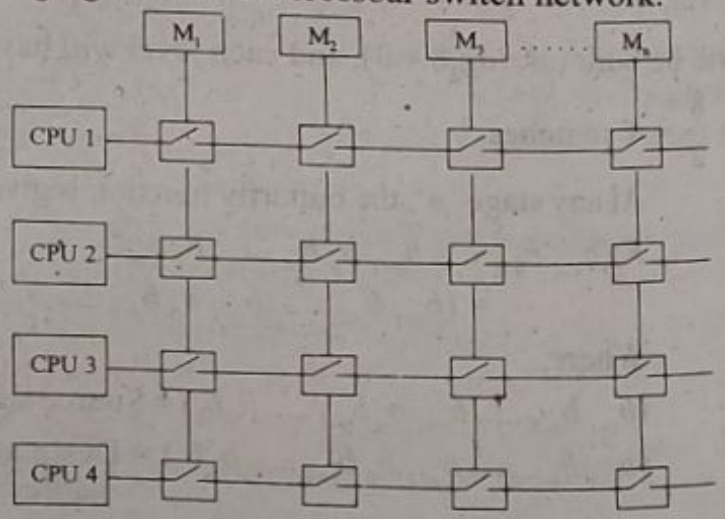


Figure (3): Crossbar Switch Network

A switch is placed at the intersection point of memory unit and a processor. By closing a switch, any processor can be connected to any memory unit. That is, a path is determined between a processor and memory by closing a switch. Hence, the switch acts as an intelligent device with some control logic so that it can set a transfer path. It identifies the memory module address placed on the bus and resolves the conflicts that occurs during the requests for accessing the same memory units, by assigning priorities to the processors etc. In a multiprocessor system, if there exists a direct connection between a processor and a memory unit, then these type of systems are often referred to as fully connected networks. On the other hand, if simultaneous transfers occur inside Q network consisting of 'n' processors and 'n' memory units then such a network is referred to as a non-blocking network. As the number of switches (n) increases, the number of crossbar switch points (n^2) also increases resulting in a very complex and complicated system.

Following is the block diagram of crossbar switch.

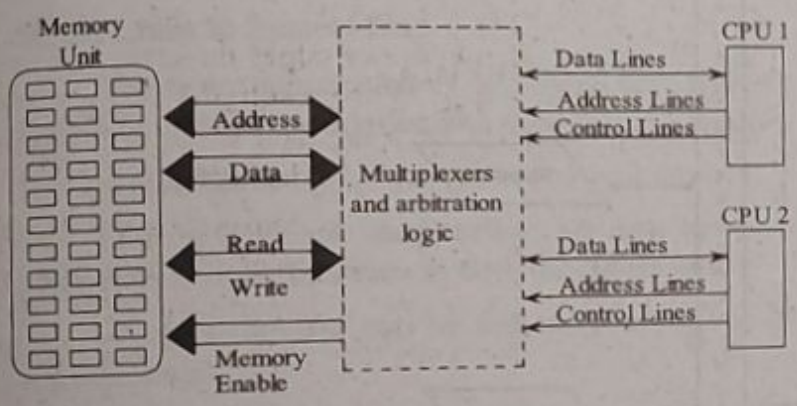


Figure (4): Block Diagram of Crossbar Switch

In the above figure, the intermediate circuitry i.e., multiplexer enables the communication between a processor and a given memory unit by selecting the required data, address and control lines initiating from that processor. Priorities are assigned to the CPUs with the help of arbitration logic and hence conflicts are resolved. The binary code which is generated internally within the arbitration logic by the priority encoder, helps in controlling the multiplexers.

Q24. Draw and explain the structure of general purpose multicomputer.

Answer : Dec.-19(R18), Q10(b)

The three shared memory multiprocessor models are as follows,

1. Uniform Memory Access (UMA)

In this model, the physical memory is equally shared among all the processors and every processor sets equal time for accessing all the memory words available. In addition to shared memory, every processor has its own private cache.

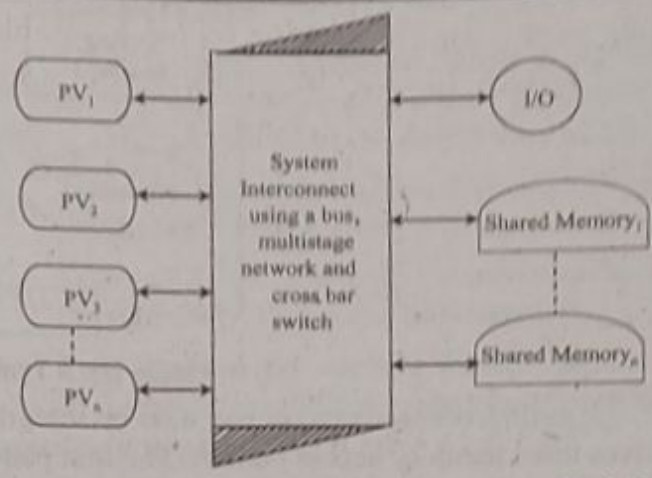


Figure (1): A Typical UMA Multiprocessor Model

Resources are shared by multiprocessors using a common bus, a crossbar switch and a multistage network. The multiprocessors that shared resources are sometimes referred as tightly coupled systems. A typical UMA multiprocessor model is shown in figure (1).

The parallel events, synchronization and inter-processor communication involved in this model are coordinated using shared variables present in the common memory.

Symmetric Multiprocessor

A system in which all the processors have uniform access to all devices of a computer is known as symmetric multiprocessor. In such systems, every processor has the capability to run, execute programs like I/O service routines and operating system kernel.

Asymmetric Multiprocessor

The system which allows only one processor to access I/O service routines and operating system kernel programs is referred as asymmetric multiprocessor. This processor is called master or executive processor and remaining processors are called attached, as they lack I/O capability. The master processor allows attached processors to execute the user code under its guidance.

Applications

(i) It is well-suited for general purpose, timesharing applications.

(ii) It is also used to enhance the execution speed of a large program in time-critical applications.

2. Non-Uniform Memory Access (NUMA) Model

In this model, the access time is not uniform as it depends on the memory location and changes accordingly. The shared memory is distributed among all processors such that every processor has its own local memory. These local memories collectively form a global address space which is allowed to be accessed by all the processors available.

The access of a local memory takes lesser time when compared to the access of a remote memory. This is because, the access to a remote memory connected to a remote processor requires additional delay caused due to interconnection network. A typical model of shared local memories is shown in figure (2).

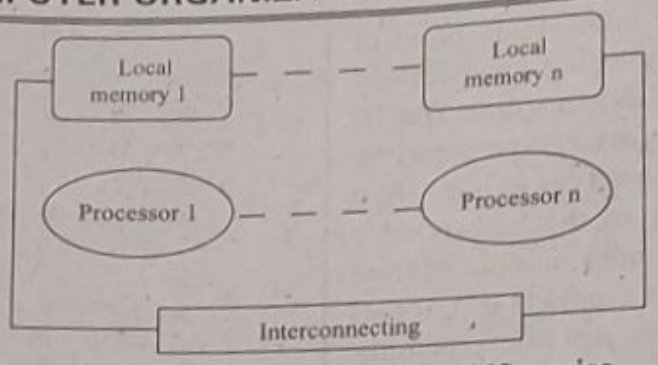


Figure (2): A Typical Shared Local Memories

A multiprocessor system can also be integrated with a globally shared memory. This shared mechanism involves three memory access pattern. The first pattern forms the local memory access which is the fastest among all. The second pattern forms the global memory access which is less faster than the first pattern. The third pattern forms remote memory access which is the slowest among all the three patterns.

Hierarchical Cluster Model of NUMA

In this model, the processors are partitioned to form multiple clusters wherein every cluster itself is either a UMA or a NUMA multiprocessor. These clusters are linked to the modules of global shared-memory and the complete system forms a NUMA multiprocessor.

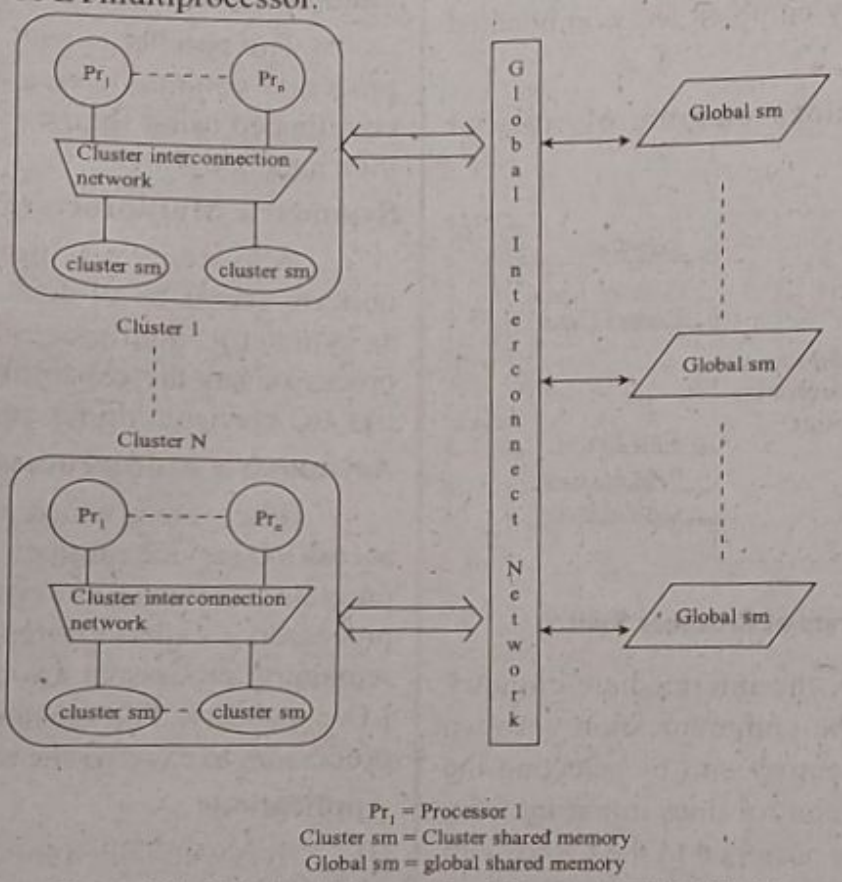


Figure (3): A Typical Hierarchical Cluster NUMA Model

In the above figure, the modules of cluster shared-memory can be seen. These modules are allowed to be accessed by all the processors associated with the same cluster. Every cluster is given equal time for accessing global memory. However, the access time of cluster memory is less when compared to that of global memory.

3. Cache Only Memory Access (COMA)

In this model , conversion of distributed main memories is done to produce caches . All these caches collectively form a global address space.

For accessing remote caches, distributed cache directories are used. The following figure depicts a typical COMA model multiprocessor.

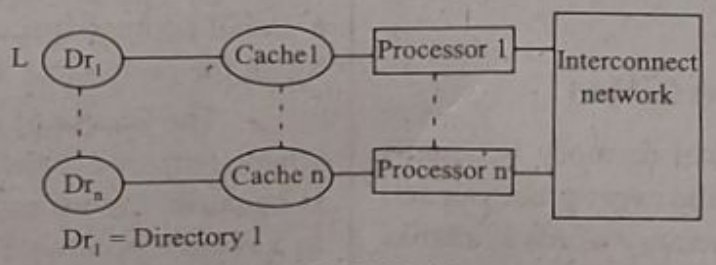


Figure (4): A Typical COMA Multiprocessor's Model

This model also makes use of hierarchical directories at times for locating the copies cache blocks.

5.3.3 Interprocessor Arbitration

Q25. What is a system bus? What are the different signal lines associated with a system bus? Explain.

Answer : **Model Paper-II, Q11(b)**

System Bus

A system bus is a special kind of bus that connects major components like memory, IOPs and CPUs in a multiprocessor system.

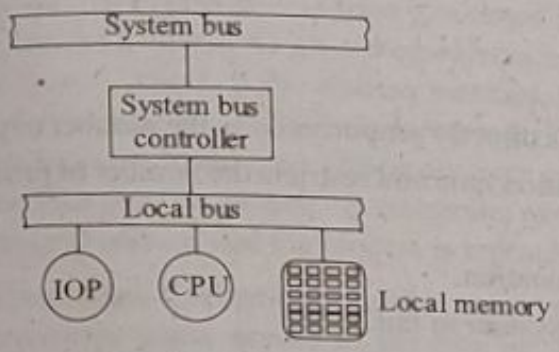


Figure: Interconnection between System Bus, Memory, IOP and CPU

Signal Lines in System Bus

There are approximately 100 signal lines within a system bus which are divided into three groups, namely,

1. Data Lines

These lines are used to transfer data between processors and the memory in both the directions.

A data transfer can be either synchronous or asynchronous.

(i) In a synchronous data transfer, a common clock source will be maintained by both the source and destination units. They agree on the same time slice and will transfer data items only during that time slice.

Alternatively, both the source and destination units can maintain separate clock sources. But, they synchronize them by sending synchronization signals at periodic intervals.

(ii) In asynchronous data transfer, the handshaking control signals are also included with the data items which are being transferred. These signals will provide information about the time when the source transferred the data and when the destination received that data.

2. Address Lines

These lines are used to determine a memory address or may be a source or destination. The total number of address lines present in a system, will define its memory capacity. That is, if there are 24 address lines in a system then its memory capacity will be 2^{24} words. These lines are unidirectional and thus transfer addresses only from processors to memory.

3. Control Lines

These lines provide special signals to control the information being transferred between processor and memory. They are two types of control lines as follows,

(i) Timing signals provide the information about the validity of data and address being transferred.

(ii) Command signals provide the information about the operations that are to be performed.

Few other signals such as transfer signals and bus control signals are also provided by control lines. Examples of transfer signals are,

(a) Memory read

(b) Memory write

(c) Transfer acknowledgment and

(d) Interrupt requests.

Examples of bus control signals are,

(a) Bus request

(b) Bus grant.

Q26. Explain the interprocessor arbitration.

April/May-23(R18), Q10(a)

OR

Discuss about the serial arbitration technique.

(Refer Only Topic: Serial(Daisy Chain) Arbitration Technique

Answer : **(Model Paper-III, Q10(a) | Aug./Sep.-22(R18), Q8(b))**

Interprocessor Arbitration

Interprocessor arbitration can be defined as a bus controller or arbitration unit that forms connections. The unit operates by forming the bus interconnections between two processors at the time when there are one or many requests for bus interconnects. Subsequently, the connection by the processor to the unit is formed by a bus called backplane bus. When the processor makes a interconnection request to unit, the unit employs a system bus and connects the processor, memory and I/O processor.

The two different techniques of interprocessor arbitration are as follows,

1. Serial (Daisy Chain) Arbitration Technique

Daisy chain connection technique provides means to achieve serial bus arbitration. It is a simple and cheap method. The below figure shows the system connections for a Daisy chain method.

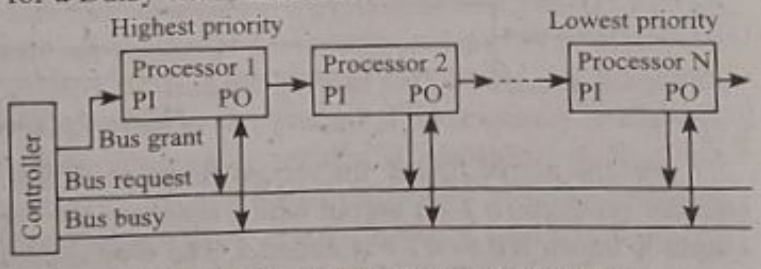


Figure: Daisy Chain (Serial) Arbitration

As shown in above figure, processors connected to the system are given priorities. Depending upon how close they are to the controller with respect to the bus grant line, the nearest processor is assigned the highest priority while the last processor is assigned with the lowest priority. Hence, this mechanism is biased towards the closer processors. All the processors make use of the same lines. So, the signal on the bus request line is the logical OR of the bus requests from all the devices connected to it. When more than one processor are trying to access the bus, and if the processor with the highest priority (i.e., the closest processor to the controller) gains access to the bus, it passes it is a requesting processor, then it blocks the propagation of the grant signal to other devices. Otherwise, it passes the grant signal to the neighbouring processor by assisting PO (Priority Out). After gaining access to the bus, the processor asserts the bus busy line to intimate other processors that the bus is currently being used.

Advantages

1. It is a simple and cheap method.
2. It requires the least number of lines.

Disadvantages

1. The propagation delay associated with the bus grant signal is directly proportional to the number of processors present in the system. This delay slow-downs the arbitration time and restricts the number of processors used in the system.
2. The priority of the processor is fixed based on its physical location.
3. Failure of any of the available processors causes the whole system to fail.

2. Parallel Bus Arbitration Technique

For answer refer Unit-V, Page No. 180, Q27.

Q27. Explain about parallel bus arbitration technique.

Answer :

Parallel Bus Arbitration Technique

In parallel bus arbitration technique, all the processors are assumed to be associated with a bus arbiter. As a parallel scheme is employed, each bus arbiter is again associated with three signal lines. They are,

1. **Bus Request:** It is an output line that will be enabled when a processor issues a request for accessing the system bus.
2. **Bus Acknowledge:** It is an input line that will be enabled when a processor gets the control over the system bus.
3. **Bus Busy:** It allows the control to be transferred in a particular order.

With this technique, only a single processor with the highest priority is selected, among all the processors. This is done with the help of an external priority encoder and a decoder.

Suppose, there are four processors, then, to implement the parallel arbitration technology, a 4 × 2 priority encoder and a 2 × 4 decoder are used. The connections between the processors, priority encoder and decoder is shown below,

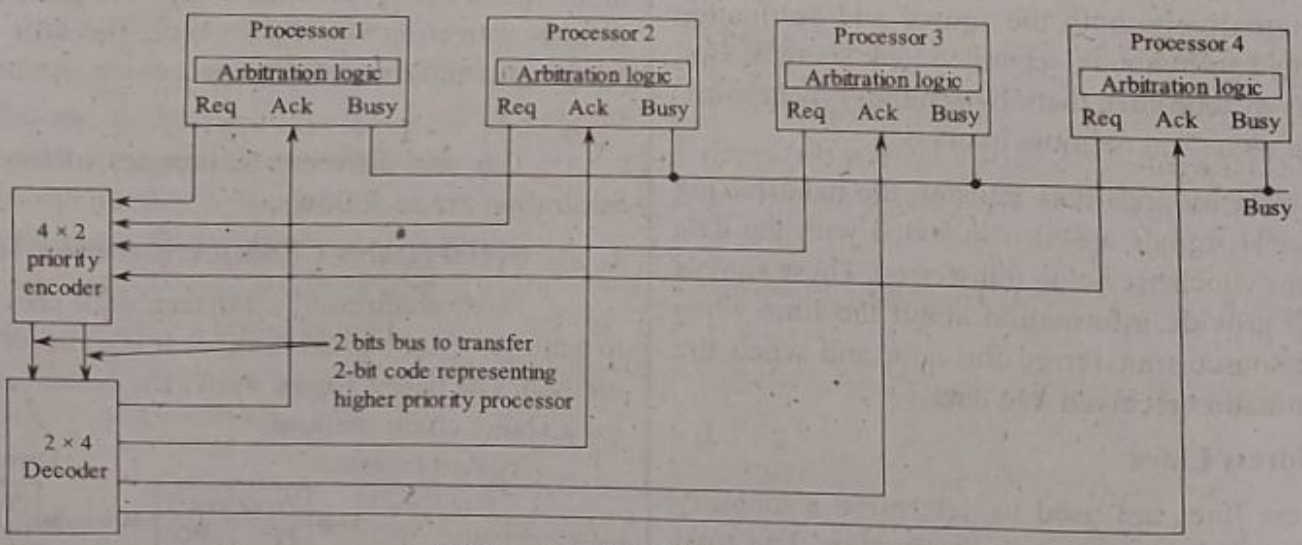


Figure: Parallel Arbitration Logic

In the above figure, the request lines of all the four bus arbiters are fed to a 4 × 2 priority encoder. This encoder produces a 2-bit output which indicates the highest priority unit among all the four processors. This 2 bit output is inturn fed to a 2 × 4 decoder. The 4-bit output from the decoder will enable the acknowledge line of the highest priority processor and the bus access is granted to it.

Q28. Discuss various dynamic arbitration techniques.

Answer :

Dynamic Arbitration Techniques

The dynamic priority allocation is a mechanism that has the ability to change the priorities of the processors while the system is in operation. Following are the various dynamic arbitration techniques.

1. **Polling Technique:** Polling technique, also known as busy/waiting technique, in which the processors repeatedly checks the address generated by the controller to see if the generated address belongs to it or to others. If the address matches, then it activates the busy line to inform other processors that the bus is being used. If there are more pending requests to be processed, the controller generates the next address and the process is repeated.
2. **Time Slice Technique:** In this scheme, each processor gains access to the bus for a fixed amount of time in a round-robin fashion. This technique is fairly unbiased, since allocated time for each processor is same, no preference is given to any particular device.
3. **LRU Technique:** LRU stands for Least Recently Used. In this scheme, a device which has not used the bus for a longer period of time is given the highest priority. Here, the priorities are assigned dynamically. Each device is given a chance to access the bus due to dynamically varying priorities.
4. **FIFO Technique:** FIFO stands for First-in-First-out. In this technique, the processor requests are served in the order in which they are received. That is, the first request is served first and the last request is served last. To implement this technique, the bus controller maintains a queue of processors requests.

5.3.4 Interprocessor Communication and Synchronization

Q29. Explain the interprocessor communication and synchronization. April/May-23(R18), Q11(a)

OR

Explain in brief inter-processor communication.

(Refer Only Topic: Interprocessor Communication (IPC))

Answer : (Model Paper-III, Q10(b) | March-22(R18), Q8(a))

Interprocessor Communication (IPC)

In a multiprocessor environment, processors achieve parallelism by doing concurrent processing. The concurrent processing requires sharing of resources between the processors. Parallelism depends on how Interprocessor Communication (IPC) is implemented. The interprocessor communication can be established through common input-output channels. Basically, the two ways by which the interprocessor communication is achieved are as follows,

(i) **IPC Using Shared Variables:** In shared memory systems, interprocess communication is achieved by using shared variables. In such systems, the shared variables are stored in the common memory which is accessible by all processors in the system. This is illustrated in the following figure,

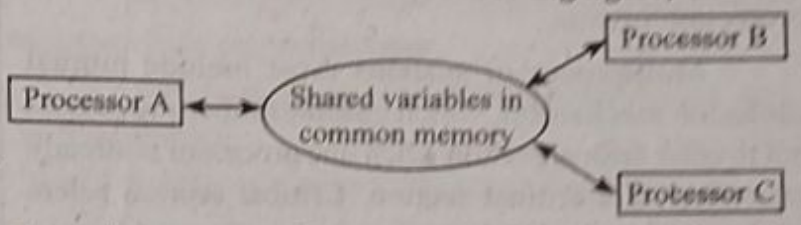


Figure: Interprocessor Communication Using Shared Variable

While sharing common resources or shared variables, conflicts/problems may arise. It is necessary to prevent conflicts with the use of shared resources by several processors. This task is done by the operating system.

(ii) **IPC Using Message Passing:** In multiprocessor systems with no shared memory, message passing mechanism is used to perform Interprocess Communication (IPC). The communication between processors is achieved by passing a message through I/O channels. This is illustrated in the following figure.

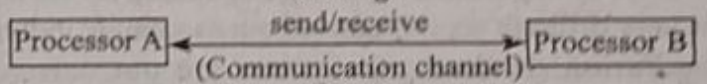


Figure: IPC Using Message Passing

When a processor wants to communicate with another processor, it uses a special procedure which initiates communication. Once the destination processor is identified, a communication channel is established. Messages are then sent or received through the communication channel.

Interprocessor Synchronization

At the higher level of exploiting parallelism, the program is partitioned into several processes that are executed on different processors. This technique is called *concurrent processing*. During concurrent processing, when two or more processes need the same resource at the same time, then contention problem arises. Such problem can be resolved by synchronization. Therefore, in multiprocessor systems, in order to synchronize concurrent processes, special synchronization software tools are used that are usually supported by hardware mechanisms. In large-scale or high contention situations, synchronization can become a performance bottleneck because contention results in additional delays. So, inorder to achieve synchronization, a set of hardware primitives are used to automatically read and modify a memory location without any interruption.

Q30. Give a brief note on mutual exclusion with a semaphore.

Answer : Sep.-21(R18), Q5(b)

Mutual Exclusion using Binary Semaphores

Mutual exclusion is the ability to share code, resources or memory in such a way that only one process can access the shared region at a time. In this mechanism, two or more processes are restricted from modifying the data at the same time thereby ensuring protection of data.

Multiprocessor systems must include mutual exclusion mechanism, which enforces other processors not to enter critical section when one processor is already executing in a critical section. Critical section refers to a segment of code that can only be executed by one processor at a time. A binary semaphore is a semaphore that can only take the values 0 and 1. It determines whether a processor is present in a critical section or not. If the value of semaphore is 1, then it indicates that processor is present in a critical section. If the value is 0, then it indicates that critical section is available to any of the requesting processor. It is necessary for all the processors to set the semaphore value to 1, whenever they are present in the critical section and must clear the value to zero or when they have completed the execution in critical section.

Testing and Setting

The operation of testing and setting the semaphore is very difficult so, it is carried out as a single indivisible operation. If these operations are not carried out independently then more than two processors try to test and set the semaphore at the same time, due to which all the processes initiate simultaneous execution of critical section. Such execution results in loss of important data and incorrect initialization of control parameters.

Initialization

A 'hardware lock' is a signal that is generated by the processor. It helps in initializing the semaphore when used with test-and-set instruction. Since, when a hardware lock signal is active, it restricts all the processors other than the one that has issued the lock signal from using the system.

When a semaphore is to be initialized, the test-and-set instruction tests and sets it and then activates the lock signal when the instruction gets executed. Thus, the semaphore cannot be modified by other processors when a processor is testing and setting it.

Suppose, SEM-ADDR is the address of the memory word whose least significant bit represents a semaphore. And, the 'test-and-set while locked' operation be represented using the mnemonic 'TSL'. Then, the instruction to execute this operation will be,

TSL SEM-ADDR

This instruction requires two uninterrupted memory cycles to get executed. The first cycle is for reading or testing the semaphore and the second cycle is for writing or setting the semaphore.

These cycles are,

First Cycle: To test or read semaphore.

R ← M[SEM-ADDR]

Second Cycle: To set or write semaphore.

M[SEM-ADDR] ← 1

In the first cycle, when the value of semaphore is moved into processor register, it tests the semaphore. If the value 1 is moved into *R* then it concludes that the semaphore is already set by another processor to execute the critical section. Thus, the new processor, which is testing the semaphore, will not be allowed to access the shared memory.

If the value 0 is moved into *R* then it concludes that the semaphore is free and the processor is allowed to access the shared memory and execute the critical section.

In the second cycle, the semaphore is set to 1 in order to restrict other processors from accessing the shared memory.

When the semaphore is set, the processor can start executing the critical section by accessing the shared resources. However, at the end of the program, the semaphore must be set to zero, so that the shared resource is released and becomes available for other processors.

5.3.5 Cache Coherence

Q31. What is cache coherence problem? Discuss about different cache coherence approaches.

Answer : Model Paper-III, Q11

Cache Coherence

Cache coherence refers to situation wherein multiple cached copies of the shared data are maintained such that all copies have the same value.

It is a known fact that, in order to balance between the speeds of a processor and the main memory, a small memory unit known as *cache* is introduced. Due to this, processor need not search the entire main memory unit for the sake of the required data, rather it refers the cache memory for satisfying the requirement. In a multiprocessor system, apart from the shared memory (the memory shared by all the processors) each processor maintain its own set of memories referred to as *local memories*. The processor can utilize either a part or the entire local memory as a cache memory.

In order to write the data to these memories, multi-processor system adopts two types of mechanisms referred to as,

- **Write Through Mechanism:** In this mechanism, whenever certain data is modified in the local memory of a processor, the same modification is also reflected in the shared memory.

- **Write Back Mechanism:** In this mechanism, whenever certain data is modified in the local memory of a processor, the same modification is not reflected in the shared memory, rather the modification process is postponed by marking the required memory location.

Hence, in order to carry out data-related computations correctly, a copy of the given data should be placed in all local memories as well as in shared memory. Maintenance of the copies of data in these multiple memories may lead to erroneous results. Hence, this problem is referred to as *cache coherence problem*.

Various Conditions for Incoherence

Cache coherence problems usually, occur when same copy of data is maintained in multiple memories. Let us view the cause and outcome of this problem through various perspectives.

Assume a multiprocessor system which consists of three processors (*P*1, *P*2, *P*3) with each processor having its own memory units referred to as *Cache*1, *Cache*2 and *Cache*3 as shown in figure (1). Apart from this configuration, the system also consists of a single large memory unit which is shared by all these processors. Imagine that, there is a variable '*L*' with a value '1' on which all the given processors are operating and this value is stored in main memory. As it is frequently required by these processors, they even load this variable along with its value in their respective cache memories. This may lead to cache coherence problem.

Cache Coherence Approaches

Assume that due to some reasons, the value of '*L*' stored in Cache2 is altered by the *P*2. This can be done by using "write through" or "write back" mechanisms.

1. **Write through Mechanism**

In this mechanism, the *P*2 alters the value stored in Cache2, this modification should also be reflected in the shared memory.

The modified data in Cache2 is reflected in shared memory but not in other cache memories. Hence, the data in Cache2 and shared memory becomes consistent but, the data present in Cache1 and Cache3 remains inconsistent.

2. **Write Back Mechanism**

In this mechanism, whenever the data in Cache2 is altered, this modification does not get reflected in the shared memory at the same time. Rather, the shared memory location is marked and this work is postponed. As a consequence, the data in Cache2 remains consistent and the data in shared memory, Cache1 and Cache3 remains inconsistent.

Outcomes

The cache coherence problem may lead to erroneous results as the entities working on consistent data will yield results that are different from the entities working on inconsistent data.

Q32. What is cache coherence problem? Discuss solutions for it.

Answer : March-21(R18), Q8(b)

Cache Coherence Problem

For answer refer Unit-V, Page No. 182, Q31.

Cache Coherence Solutions

Cache coherence solutions are of two types. They are as follows,

1. **Software Cache Coherence Solution**

The software based solution for handling cache coherence is the use of operating system and compiler. The compiler performs the following function.

(a) It analyzes the code.

(b) It determines the data items that can be harmful for caching.

(c) It marks the identified harmful items.

(d) It ensures that the marked items are not cached.

Shared data variables should also be protected from being cached because cache coherence issue may arise if these variables are cached. These shared variables, when needed by more than one process at same time may be updated by one process while other process might need it. Moreover, there are some software cache coherence solutions which identify safe periods for the shared data variables by analyzing the variables. These variables are providing instructions for maintaining cache coherence.

Advantages of Software Solution

- Additional hardware circuitry and logic needed by hardware approach is avoided in software based approach.
- Cache coherence detection problem is transferred to compile time from run time.
- The design issues are also handled by the software based approach.

Disadvantages of Software Solution

- The cache utilization efficiency is reduced as the software approach transfers the handling issue to compile time.

2. **Hardware Cache Coherence Solution**

Small scale multiprocessors are employing many hardware solutions for maintaining coherence cases. Since, usage of migration and replication properties prove to be a compromise with the performance criteria. The hardware solution protocols called cache coherence protocols have capability of maintaining multiprocessor coherence. There are two types of cache coherence protocols namely,

(a) Snooping Protocol

This protocol is generally used by shared memory multiprocessors architecture, where in cache memories are connected to a common bus, which in turn is connected to the common single shared memory. The snooping protocol requires a constant connection between the individual caches and the common shared memory. Since, in shared memory architecture the common bus which connects the array of multiprocessors to the common shared memory for data transfer proves to be an effective medium to implement cache coherence using snooping protocol. Without any further hardware additions, snooping protocol makes use of the common bus as a medium for snooping i.e., placing the address on the bus so that processors communicate and try to match the address with their respective caches.

Snooping protocols are categorised into two types each of which are used for maintaining the coherence property.

(i) Write invalidate protocol

(ii) Write update protocol.

(b) Directory based Protocol

Directory based protocol is an alternative to snooping protocol and is used mostly on distributed shared-memory architecture, which is specific to large scale processor. A directory maintains the information about cached block state. In addition to this, a directory also maintains the information about,

- The list of all caches that have copies of a particular memory block.
- Block is dirty (i.e., informally read (or) data written on it) or not.

The structure of the directories is changing overtime in order to reserve bandwidth and reduce access latencies. The present directories are made in such a way that, they store information of cache memories but not main memory. There are other directories that are designed in a way, that they store data memory (cache as well as main) upto fixed bits (bits per entry scheme). It is necessary to distribute the directory entries which specify the location of a memory across the memories so as to avoid directory related issues. This helps different directory entries pointing to different memory locations. Each directory entry points to only one known memory location due to which broadcast can be avoided by using coherence protocol.

A directory can be connected to a common bus (or) might have a separate port to the memory (or) can be a part of node controller which is responsible for inter node and intranode communication.

VERY SHORT QUESTIONS WITH SOLUTIONS (VSQS)

Q1. Define speedup ratio.

Answer :

Model Paper-III, Q1(i)

Speedup ratio is given by,

$$s_u = \frac{mt_m}{(c+m-1)t_p}$$

Where,

m – Number of tasks

c – Number of segments in the processor with pipelining

t_m – Time taken to complete a task in a nonpipelined processor

t_p – Time taken to complete 'm' tasks in a pipelined processor.

In this equation, the numerator corresponds to the processor without pipelining. It takes time 't_m' to complete a single task and since there are 'm' tasks the total time taken for their completion is 'mt_m'.

Q2. List two advantages of pipeline processing?

Answer :

Model Paper-I, Q1(i)

1. Complex arithmetic and logic units can be designed in less time with the pipelined processors.
2. High performance of CPU can be achieved by working with higher clock frequency.

Q3. What do you mean by super computer?

Answer :

Model Paper-II, Q1(i)

The computers that combine the features of vector processing along with the floating-point computations are generally referred to as super computers.

Q4. Write a short note on CISC.

Answer :

CISC (Complex Instruction Set Computing) is another design strategy which consists of full set of computer instructions. These instructions are responsible for providing the required capabilities efficiently. Thus, instruction-set consists of 120 to 350 instructions.

Q5. Write in brief about RISC.

Model Paper-II, Q1(j)

Answer :

The RISC processors instruction sets are smaller in size and includes nearly 100 instructions within it. These instructions must carryout wide range of functions and must be able to control any type of task.

Q6. Define superscalar processor.

Answer :

A processor that has multiple processing units each executing one instruction at a time is called a superscalar processor.

Q7. Write a short note on cache coherence.

Model Paper-I, Q1(j)

Answer :

Cache coherence refers to situation wherein multiple cached copies of the shared data are maintained such that all copies have the same value.

Q8. What do you mean by snooping protocol.

Model Paper-III, Q1(j)

Answer :

This protocol is generally used by shared memory multiprocessors architecture, where in cache memories are connected to a common bus, which in turn is connected to the common single shared memory. The snooping protocol requires a constant connection between the individual caches and the common shared memory.

FREQUENTLY ASKED QUESTIONS & IMPORTANT QUESTIONS

Q1. What is parallel processing? Explain Flynn's classification of computer.

(Important Question | March-21(R18), Q7(a))

Answer :

For answer refer Unit-V, Page No. 160, Q4.

Q2. Explain four segment instruction pipeline.

(April/May-23(R18), Q10(b) | Aug./Sep.-22(R18), Q8(a))

Repeated 2 Times

Answer :

For answer refer Unit-V, Page No. 164, Q9.

Q3. Discuss the various conflicts that might arise in a pipeline. How are they resolved?

(Important Question | Dec.-19(R18), Q10(a))

Answer :

For answer refer Unit-V, Page No. 166, Q11.

Q4. Discuss about RISC Pipeline.

(Important Question | March-21(R18), Q8(a))

Answer :

For answer refer Unit-V, Page No. 166, Q12.

Q5. Illustrate vector operations and vector processing.

(Important Question | March-21(R18), Q7(b))

Answer :

For answer refer Unit-V, Page No. 167, Q14.

Q6. Explain the array processors.

(Important Question | April/May-23(R18), Q11(b))

Answer :

For answer refer Unit-V, Page No. 169, Q16.

Q7. Discuss the characteristics of multi-processors.

(Important Question | March-22(R18), Q8(b))

Answer :

For answer refer Unit-V, Page No. 171, Q18.

Q8. Draw and explain the structure of general purpose multicomputer.

(Important Question | Dec.-19(R18), Q10(b))

Answer :

For answer refer Unit-V, Page No. 177, Q24.

Q9. Explain the interprocessor arbitration.

(April/May-23(R18), Q10(a) | Aug./Sep.-22(R18), Q8(b))

Repeated 2 Times

Answer :

For answer refer Unit-V, Page No. 179, Q26.

Q10. Explain the interprocessor communication and synchronization.

(April/May-23(R18), Q11(a) | March-22(R18), Q8(a))

Repeated 2 Times

Answer :

For answer refer Unit-V, Page No. 181, Q29.

Q11. Give a brief note on mutual exclusion with a semaphore.

(Important Question | Sep.-21(R18), Q5(b))

Answer :

For answer refer Unit-V, Page No. 182, Q30.

Q12. What is cache coherence problem? Discuss solutions for it.

(Important Question | March-21(R18), Q8(b))

Answer :

For answer refer Unit-V, Page No. 183, Q32.