Unit 4

Q1) Instruction formats :-

An instruction format or instruction code is a group of bits used to perform a particular operation on the data stored in computer. Processor fetches an instruction from memory and decodes the bits to execute the instruction. Different computer may have their own instruction set.

The operation of the processor is determined by the instructions it executes, referred to as machine instructions or computer instructions. The collection of different instructions that the processor can execute is referred to as the processor's instruction set.

Each instruction must contain the information required by the processor for execution.

Elements of a Machine Instruction

1) Operation code (opcode) :- Specifies the operation to be performed. The operation is specified by a binary code known as the operation code, opcode do ADD, SUB, DIV, LOAD....

2) Source operand reference :- The operation may involve one or more source operands.

that is, operands that are inputs for the operation.

3) Result operand reference :- The operation may produce a result

4) Next instruction reference :- This tells the processor where to fetch the next instruction after the execution of this instruction is complete.

Opcode-field	Address-field

Instruction format

All the temporary data can be stored either in the main memory or register or can directly be sent to the input-output device. Thus, the address field can be :

- Main or Virtual Memory :- It contains the memory address of the main memory.
- CPU Register :- CPU contains one or more registers that may be referenced by machine instructions. If only one register exists, reference to it may be implicit. If more than one register exists, then each register is assigned a unique number, and the instruction must contain the number of the desired register.
- Input/Output Device :- The instruction must specify the I/O module or device for the operation. The memory-mapped I/O can be used for storing or retreiving instructions from certain memory address.

Q2) Instruction Set :- The instruction set, also

called instruction set architecture (ISA), is part of a computers that pertains to programming, which is basically machine language. The instruction set provided provides commands to the processor, to tell it what it needs to do. The instruction set consists of addressing modes, instructions, native data types, registers, memory architecture, interrupt and exception handling, and external I/O.

An example of an instruction set is the x86 instruction set, which is common to find on computers today. Different computer processors can use almost the same instruction set while still having very different internal design. Both the Intel Pentium and AMD Athlon processors use nearly the same x86 instruction set. An instruction set can be built into the hardware of the processor, or it can be emulated in software using an interpreter. The hardware design is more efficient and faster for running programs than the emulated software version.

Examples of instruction set

1) ADD :- Add two numbers together
2) COMPARE :- Compare numbers
3) IN :- Input information from a device eg - Keyboard
4) JUMP :- Jump to designated RAM address.
5) JUMP IF :- Conditional statement that jumps to a designated RAM address.
6) LOAD :- Load information from RAM to the CPU.
7) OUT :- Output information to device eg :- monitor

Camlin Page
Date / /

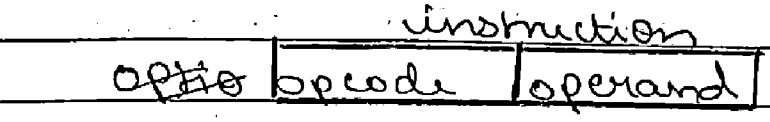
8) STORE: Store information to RAM.

Q.37 Addressing Modes:- The addressing modes in computer architecture actually define how an operand is chosen to execute an instruction. It is the way that is used to identify the location of an operand which is specified in an instruction.
whenever an instruction executes, it requires operands to be operated on. An instruction field consists of opcode and operand. Where operand means the data and opcode means the instruction itself. In case of operations like addition or subtraction, they require two data. So, they are called binary instruction. On the other hand, the increment or decrement operations need only one data and are so called unary instruction.
The various addressing modes in computer architecture can be classified as below.

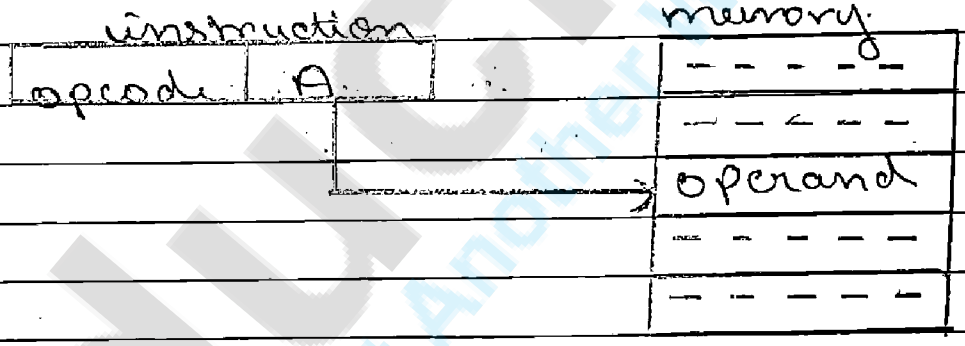
1) Implicit addressing mode:- The term implicit addressing mode means here we are not mentioning clearly in details that from where the instruction can get the operand.

2) Immediate addressing mode:- In the immediate addressing mode, the instruction contains two fields. One for the opcode and another field contains the operand itself. That means in this addressing mode, there is no need to go anywhere to access the operand because of the instruction itself containing the

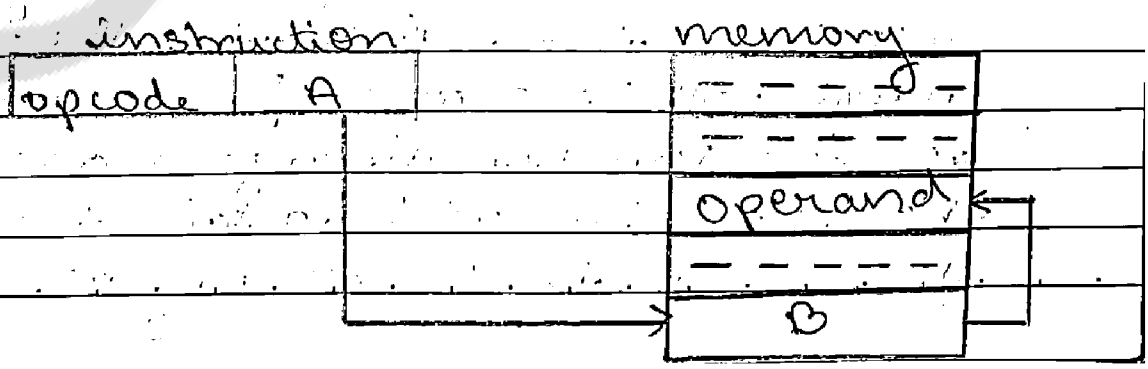
operand. This is known as immediate addressing mode.



3) Direct addressing mode: In direct addressing mode, the instruction will have the two parts. One part will contain the opcode and another one will contains the address of the memory location at where the operand can be found. Here A is the address of the operand. That means at the Ath location in the memory, the operand can be found.

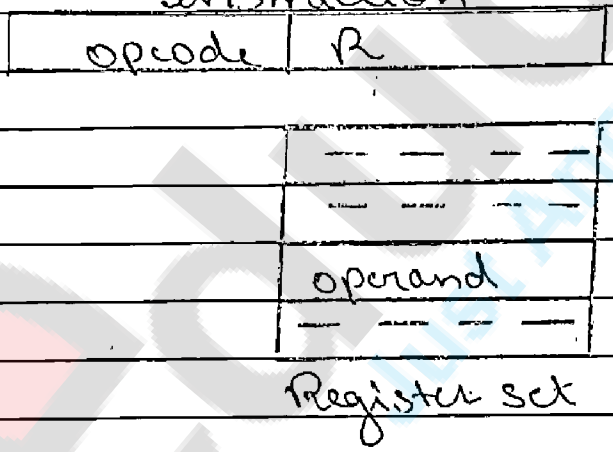


4) Indirect addressing mode: Indirect addressing mode contains the opcode and address field. But unlike direct addressing mode, it doesn't contains the address of the operand but contains the address of a memory location in which the actual address of the operand can be found.



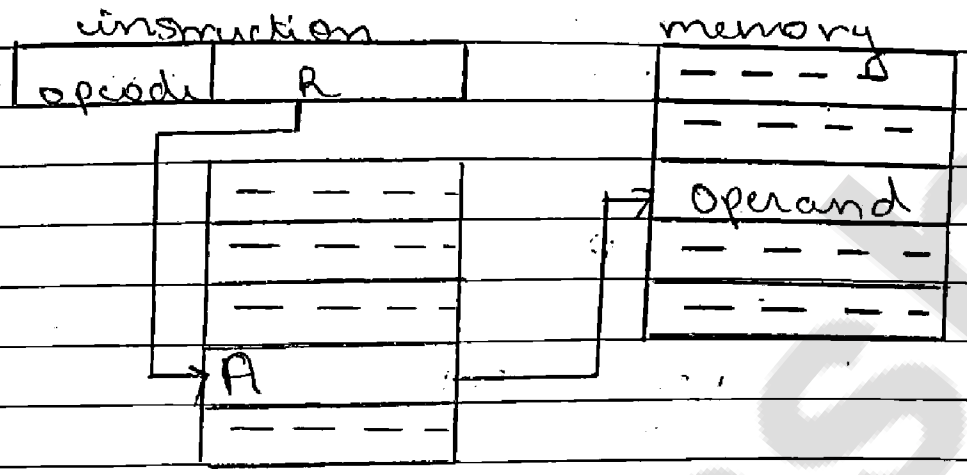
Here A contains the address of the location B in memory and B contains the actual address of the operand in memory.

5) Register addressing mode :- In case of register addressing mode, the instruction will have the opcode and a register number. Depending upon the register number, one of the registers will be selected from the available set of registers by default automatically. The unique identification of the register can be done by the register number which is mentioned in the instruction. In that register the operand can be found.



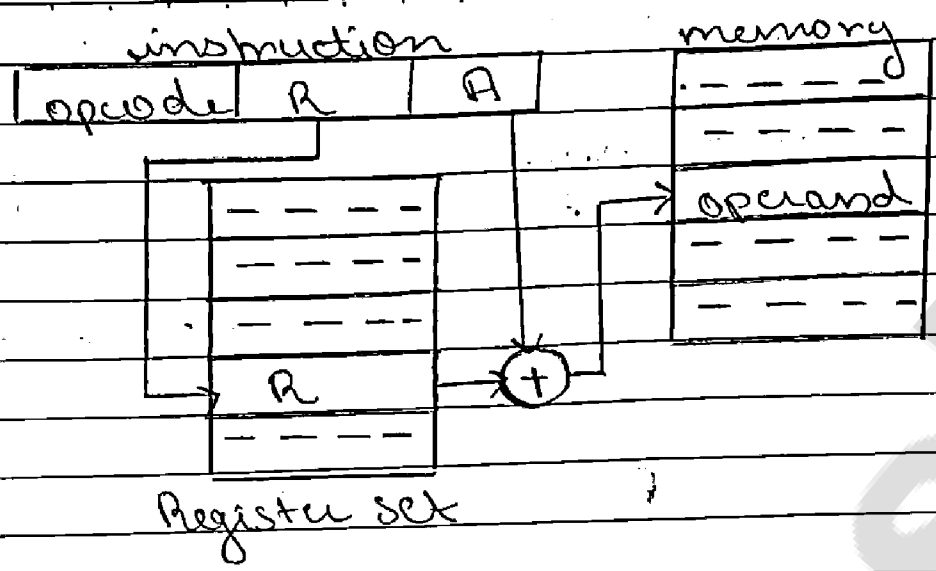
6) Register indirect addressing mode :- In the register indirect addressing mode, the instruction will contain the opcode as well as a register number. Depending upon the register number mentioned in the instruction, the corresponding register will be accessed from the set of registers. But here the register doesn't contain the operand but will contain the address of the operand in the memory at where the

operand can be found



Suppose in memory, the operand is in the A^{th} location. Now, this address A will be stored in the register and the register number say R will be mentioned in the instruction. This is called register addressing mode.

7] Displacement addressing mode :- In the displacement addressing mode, the instruction will be having three fields. One for the opcode one for the register number and the remaining one for an absolute address.
At first, depending upon the register number the register will be selected from the register set. After that that its content will be added with the absolute address and the new address formed will be the actual physical address of the operand in the memory



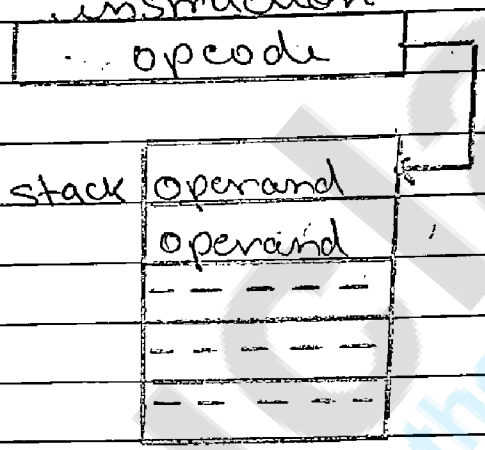
Displacement addressing mode in computer architecture can be categorized into 3 different modes.

1) Relative
2) Base register
3) Indexing

- In case of relative addressing mode, the register used will be a program counter.
- In base register addressing mode, the content of the base register is added to the direct address part of the instruction to obtain the effective address. Means, in it the register indirect address field point to the base register and to obtain effective address, the contents of instruction register is added to direct address part of the instruction.
- In case of Indexing mode, the absolute field will contain the starting base address of the memory block and the register field will contain the index value. Adding both will give the actual physical address of the operand.

Camlin | Page
Date / /

8) Stack addressing mode:- In case of stack addressing mode, the instruction knows the topmost data should be the operand. If the instruction is a unary instruction then it will select the topmost data as the operand and if the instruction is a binary instruction then it will select the topmost two data as the operands from the top of the stack.



These are the basic and primary addressing modes in computer architecture apart from these modes, we have some other addressing modes in computer architecture including auto-increment, auto-decrement mode. But above mentioned are the most important addressing modes in computer architecture or computer organization.

Q4) Processor Organization:-

Requirements placed on the processor

- Fetch instruction:- The processor reads an instruction from memory (register, cache, main memory).
- Interpret instruction:- The instruction is.

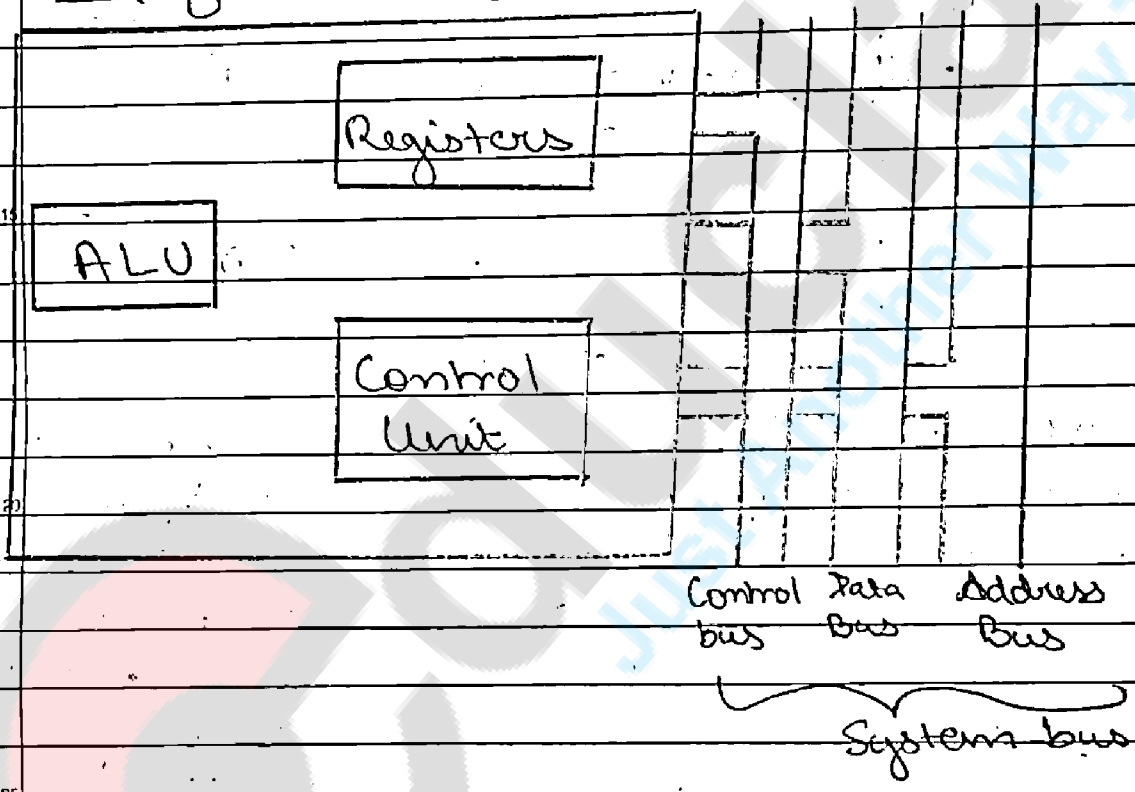
decoded to determine what action is required.

-Fetch data :- The execution of an instruction may require reading data from memory or an I/O module.

-Process data :- The execution of an instruction may require performing some arithmetic or logical operation on data.

-Write data :- The results of an execution may require writing data to memory or I/O module.

Simplified view of processor



Components of Processor

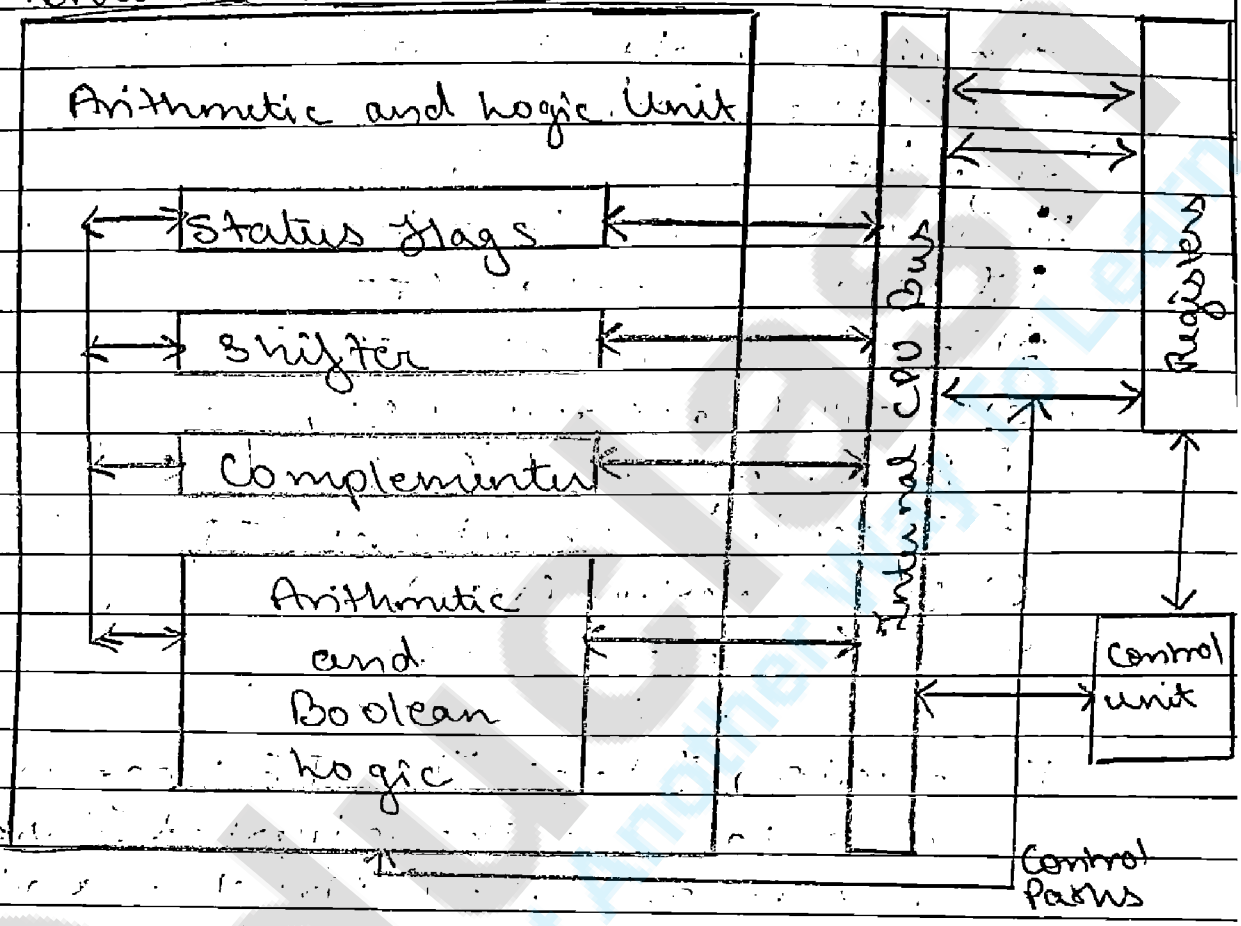
The major components of the processor are an arithmetic and logic unit (ALU) and a control unit (CU).

The ALU does the actual computation or processing of data.

The control unit controls the movement of data and instructions into and out of

the processor and controls the operation of the ALU.
Register consists of a set of storage locations.

CPU Internal Structure



The data transfer and logic control paths are indicated, including an element labeled internal processor bus.
This element is needed to transfer data between the various registers and the ALU because the ALU in fact operates only on data in the internal processor memory.

Q5) Instruction Pipeline :- In this a steam of instructions can be executed by overlapping fetch, decode and execute phases of an instruction cycle. This type of technique is used.

to increase the throughput of the computer system. An instruction pipeline reads instruction from the memory while previous instructions are being executed in other segments of the pipeline. Thus we can execute multiple instructions simultaneously. The pipeline will be more efficient if the instruction cycle is divided into segments of equal durations.

Instruction pipeline has six operations,

1) Fetch instruction (FI)
2) Decode instruction (DI)
3) Calculate operands (CO)
4) Fetch operands (FO)
5) Execute instructions (EI)
6) Write result (WR).

Overlap these operations

Instruction Fetch :- The IF stage is responsible for obtaining the requested instruction from memory. The instruction and the program counter are stored in the register as temporary storage.

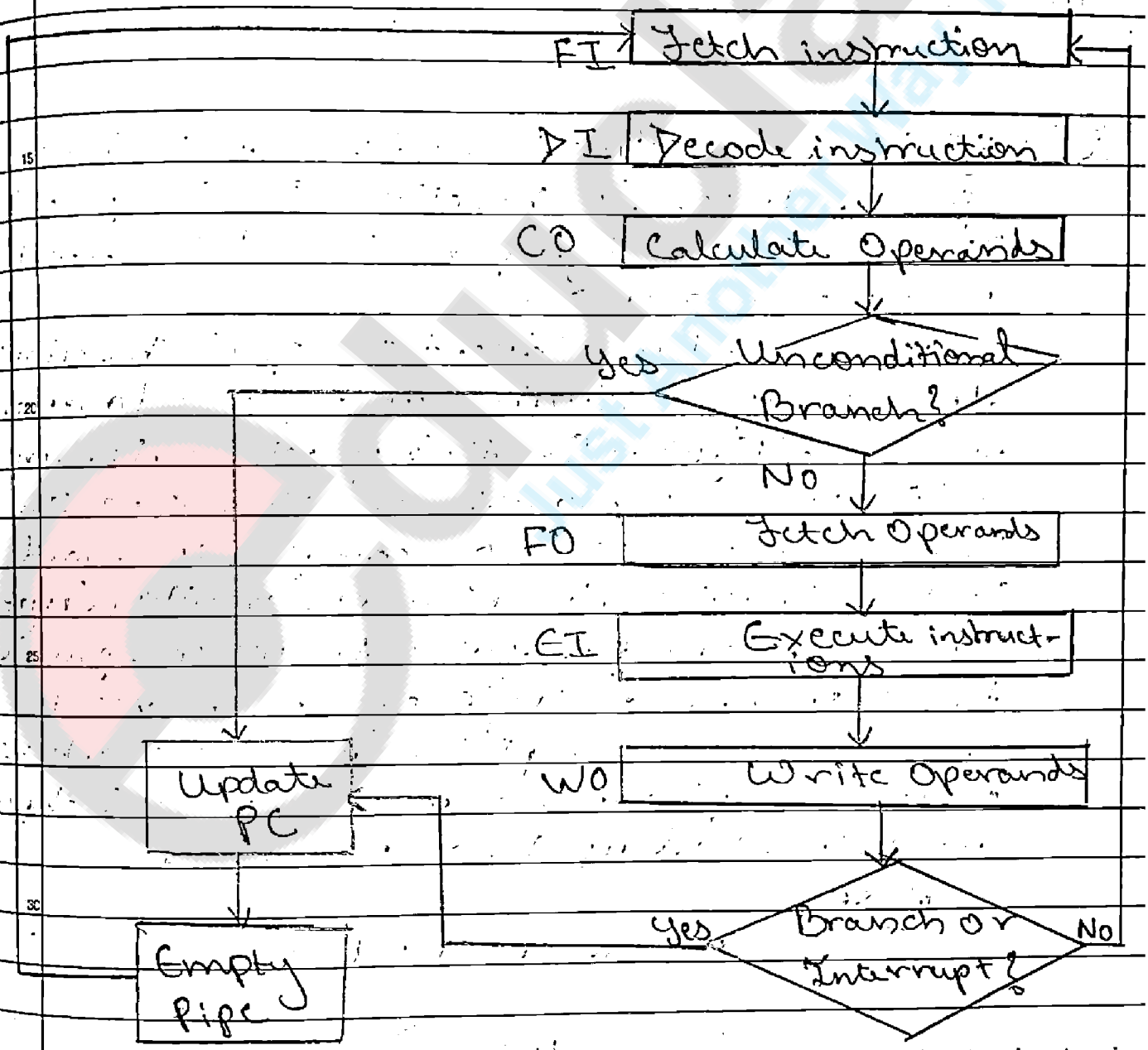
2) Decode Instruction :- The DI stage is responsible for decoding the instruction and sending out the various control lines to the other parts of the processor.

3) Calculate Operands :- The CO stage is where any calculations are performed. The main component in this stage is the ALU. The ALU is made up of arithmetic, logic and capabilities.

4) Fetch Operands and Execute Instruction:- The FO and EI stages are responsible for storing and loading values to and from memory. They are also responsible for input and output from the processor respectively.

5) Write Operands:- The WO stage is responsible for writing the result of a calculations, memory access or input into the register file.

Six Stage instruction Pipeline:-



Camlin Page
Date / /

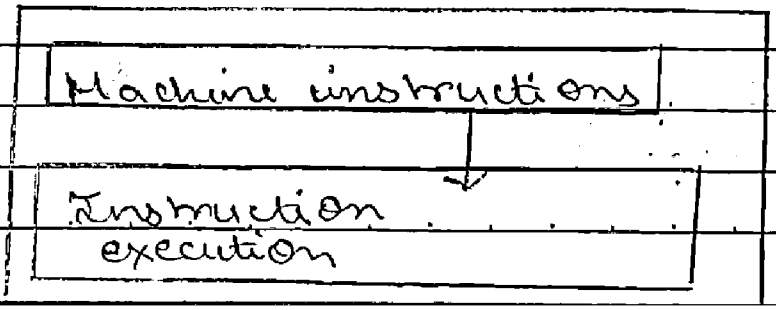
Timing Diagram for Instruction Pipelining Operation

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Instruction 1	FI	DI	CO	FO	EI	WO								
Instruction 2		FI	DI	CO	FO	EI	WO							
Instruction 3			FI	DI	CO	FO	EI	WO						
Instruction 4				FI	DI	CO	FO	EI	WO					
Instruction 5					FI	DI	CO	FO	EI	WO				
Instruction 6						FI	DI	CO	FO	EI	WO			
Instruction 7							FI	DI	CO	FO	EI	WO		
Instruction 8								FI	DI	CO	FO	EI	WO	
Instruction 9									FI	DI	CO	FO	EI	WO

Q(c) **RISC Architecture:-** The term RISC stands for "Reduced Instruction Set Computer". It is a CPU design plan based on simple orders and acts fast.

This is small or reduced set of instructions. Here, every instruction is expected to attain very small jobs. In this machine, the instruction sets are modest and simple, which help in comprising more complex commands. Each instruction is about the similar length; these are wound together to get compound tasks done in a single operation. Most commands are completed in one machine cycle. This pipelining is a crucial technique used to speed up RISC machines

RISC



RISC is a microprocessor that is designed to carry out few instructions at the similar time. Based on small commands, these chips need fewer transistors, which make the transistors inexpensive to design and produce. The features of RISC include the following:

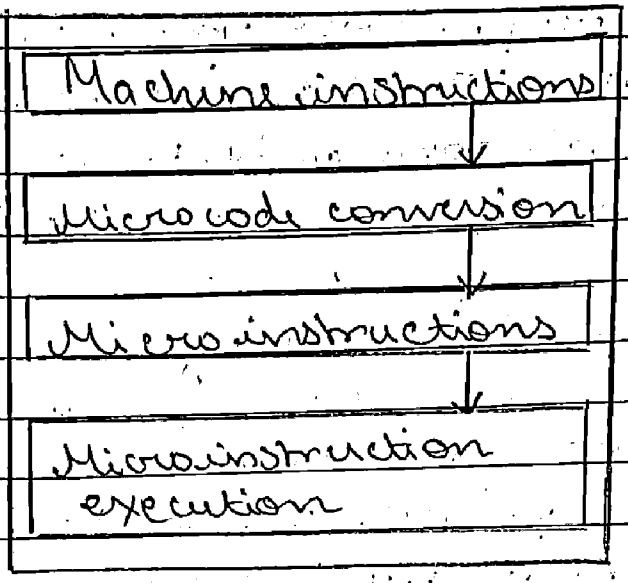
- The demand of decoding is less
- Few data types in hardware
- General purpose register is identical
- Uniform instruction set
- Simple addressing modes.

Also while writing a program, RISC makes it easier by letting the computer programmer to eliminate needless codes and stops wasting of cycles.

Q7) CISC Architecture :- The term CISC stands for "Complex Instruction Set Computer". It is a CPU design plan based on single commands, which are skilled in executing multi-step operations.

CISC computers have small programs. It has a huge number of compound instructions, which takes a long time to perform. Hence, a single set of instruction is protected in several steps; each instruction set has additional than 300 seperate instructions. Maximum instructions are finished in two to ten machine cycles. In CISC, instruction pipelining is not easily implemented.

CISC



CISC Architecture

The CISC machines have good acts, based on the overview of program compilers; as the range of innovative instructions are simply obtainable in one instruction set. They design compound instructions in the single, simple set of instructions. They achieve low-level processes, that makes it easier to have huge addressing modes and additional data types in the hardware of a machine. But, a CISC is considered less efficient than RISC, because of its incompetence to eliminate codes which lead to wasting of cycles. Also, microprocessor chips are difficult to understand and program for; because of the complexity of the hardware.

Q8) Comparison between RISC and CISC:

	RISC	CISC
Acronym	It stands for "Reduced Instruction Set Computer"	It stands for "Complex Instruction Set Computer"

Definition	The RISC processors have a smaller set of instructions with few addressing modes	The CISC processors have a larger set of instructions with many addressing modes.
Memory Unit	It has no memory unit and uses a separate hardware to implement instructions.	It has a memory unit to implement complex instructions.
Program Design	It has a hard-wired unit of programming. It is a complex compiler design.	It has a micro-programming unit. It is an easy compiler design.
Calculations	The calculations are faster and precise	The calculations are slow and precise
Decoding	Decoding of instruction is simple.	Decoding of instructions is complex.
Time	Execution time is very less.	Execution time is very high.
External memory	It does not require external memory for calculations.	It requires external memory for calculations
Pipelining	Pipelining does function correctly.	Pipelining does not function correctly.
Stalling	Stalling is mostly reduced in processors.	The processors often stall.
Code expansion	Code expansion can be a problem	Code expansion is not a problem.
Disk space	The space is saved	The space is wasted
Applications	Used in high end applications such as video processing, telecommunications and image processing.	Used in low end applications such as security systems, home automations etc.

Q7] Registers and its organization:-
Register is a very fast computer memory, used to store data/instruction in execution.
A Register is a group of flip-flops with each flip-flop capable of storing one bit of information. An n-bit register has a group of n flip-flops and is capable of storing binary information of n-bits.
A register consists of a group of flip-flops and gates. The flip-flops hold the binary information and gates control when and how new information is transferred into a register.
Following are some commonly used registers:

1) Accumulator :- This is the most common register, used to store data taken out from the memory.

2) General purpose Registers :- This is used to store data ~~taken out from the memory~~. intermediate results during program execution. It can be accessed via assembly programming.

3) Special Purpose Registers :- Users do not access these registers. These registers are for computer system.

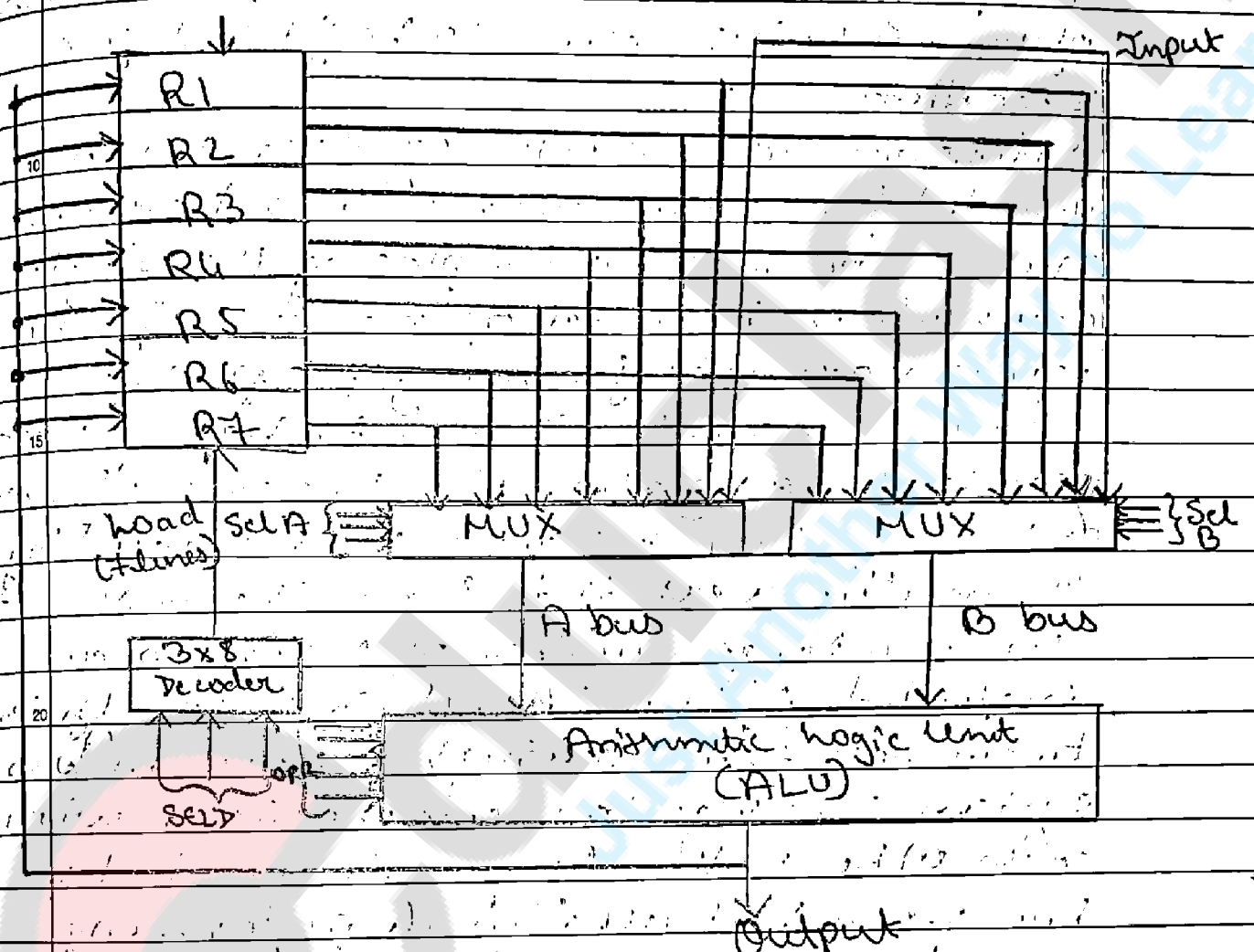
a) MAR : Memory Address Register are those registers that holds the address for memory unit.

b) MBR : Memory Buffer Register stores ~~information~~ instruction and data received from the memory and sent from the memory.

c) PC: Program Counter points to the next instruction to be executed.

d) IR: Instruction Register holds the instruction its to be executed.

A bus organization for seven CPU register



The number of registers in a processor unit may vary from just one processor register to as many as 64 registers or more.

1) One of the CPU registers is called as an accumulator AC or 'A' register. It is the main operand register of the ALU.

2) The data register (DR) acts as a buffer between the CPU and main memory. It is used as an input operand register with

the accumulator.
3) The instruction register (IR) holds the opcode of the current information.
4) The address register (AR) holds the address of the memory in which the operand resides.
The program counter (PC) holds the address of the next instruction to be fetched for execution.
Additional addressable registers can be provided for storing operands and address. This can be viewed as replacing the single accumulator by a set of registers. If the registers are used for many purpose, the resulting computer is said to have general register organization. In the case of processor registers, a register is selected by the multiplexers that form the buses.
When a large number of registers are included in the CPU, it is more efficient to connect them through a common bus system. The registers communicate with each other not only for direct data transfers, but also while performing various micro-operations. Hence it is necessary to provide a common unit that can perform all the arithmetic, logic and shift micro-operation in the processor.

Q10) Explain in detail about the different superscalar instruction issue policies.
The processor must be able to identify instruction level parallelism and

orchestrate the fetching, decoding and execution of instruction in parallel. Instruction issue refer to the process of initiating instruction execution in the processor's functional units and the term instruction issue policy refer to the protocol used to issue instructions.

In general we can say that instruction issue occurs when instruction moves from the decode stage of the pipeline to the first execute stage of the pipeline. In essence, the processor is trying to look ~~around~~ ahead of the current point of execution to locate instructions that can be brought into the pipeline and executed. Three types of orderings are important in this regard:

- The order in which instructions are fetched
- The order in which instructions are executed
- The order in which instructions update the contents of register and memory locations

In general terms, we can group superscalar instruction issue policies into the following categories:

- In-order issue with in-order completion
- In-order issue with out-of-order completion
- Out-of-order issue with out-of-order completion

1) IN-ORDER ISSUE WITH IN-ORDER

Camlin Page
Date / /

COMPLETION :- The simplest instruction issue policy is to issue policy instructions in the exact order that would be achieved by sequential execution (in-order issue) and to write results in the same order (in-order completion). Not even scalar pipelines often follow such a simple-minded policy. However, it is useful to consider this policy as a baseline for comparing more sophisticated approaches.

2) IN-ORDER ISSUE WITH OUT-OF-ORDER COMPLETION :- Out-of-order completion is used in scalar RISC processors to improve the performance of instructions that require multiple cycles. With out-of-order completion, any number of instructions may be in the execution stage at any one time, up to the maximum degree of machine parallelism across all functional units. Instruction issuing is stalled by a resource conflict, a data dependency, or a procedural dependency.

3) OUT-OF-ORDER ISSUE WITH OUT-OF-ORDER COMPLETION :- With in-order issue, the processor will only decode instructions up to the point of a dependency or conflict. No additional instructions are decoded until the conflict is resolved. As a result, the processor cannot look ahead of the point of conflict to subsequent

instructions that may be independent of those already in the pipeline and that may be usefully introduced into the pipeline.

To allow out-of-order issue, it is necessary to decouple the decode and execute stages of the pipeline. This is done with a buffer referred to as an instruction window. With this organization, after a processor has finished decoding an instruction, it is placed in the instruction window. As long as this buffer is not full, the processor can continue to fetch and decode new instructions. When a functional unit becomes available in the execute stage, an instruction from the instruction window may be issued to the execute stage. An instruction may be issued, provided that 1) it needs the particular functional unit that is available, and 2) no conflicts or dependencies block this instruction.

The result of this organization is that the processor has a lookahead capability allowing it to identify independent instructions that can be brought into the execute stage. Instructions are issued from the instruction window with little regard for their original program order. As before, the only constraint is that the program execution behaves correctly.

One common technique that is used

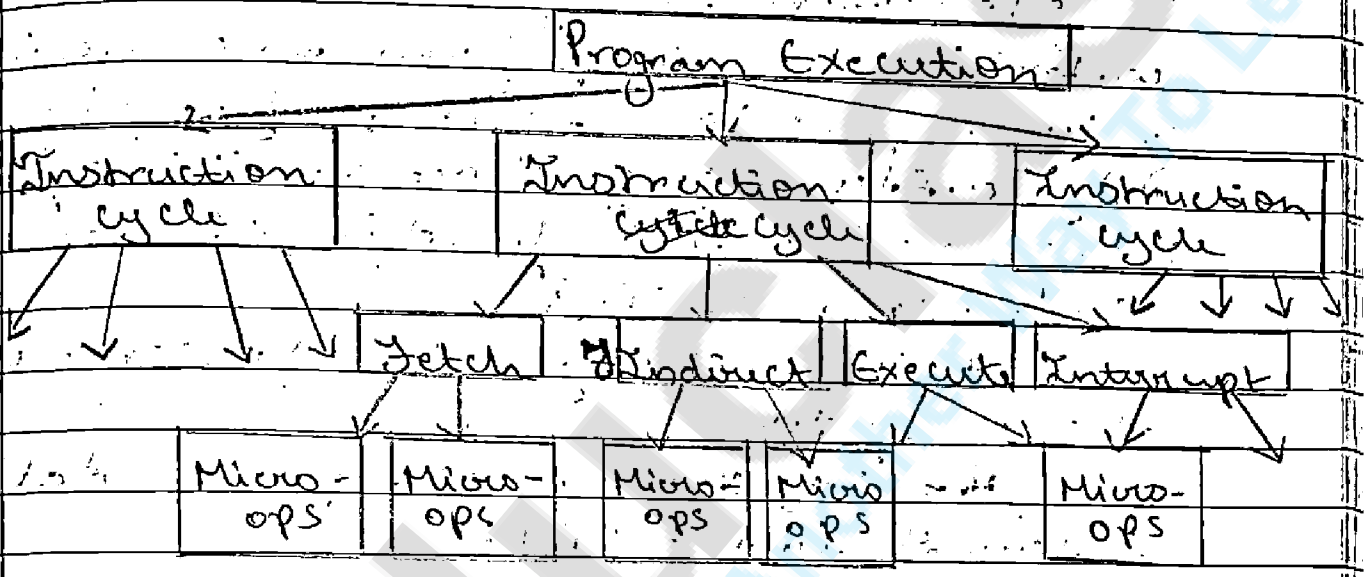
to support out-of-order completion is the reorder buffer. The reorder buffer is temporary storage for results completed out of order that are then committed to the register file in program order. ~~It is stated to~~ The term antidependency is used because the constraint is similar to that of a true data dependency, but reversed. Instead of the first instruction producing a value that the second instruction uses, the second instruction destroys a value that the first instruction uses.

Unit 5

Q1) **Micro-operations :-** In computer central processing unit, micro-operations (also known as micro-ops) are the functional or atomic operations of a processor. These are low-level instructions used in some designs to implement complex machine instructions. They generally perform operations on data stored in one or more registers. They transfer data between registers or between external buses of the CPU, also perform arithmetic and logical operations on registers.

In executing a program, operation of a computer consists of a sequence of instruction cycles, with one machine instruction per cycle. Each instru-

tion cycle is made up of a number of smaller units - Fetch, Indirect, Execute and Interrupt cycles. Each of these cycles involves series of steps, each of which involves the processor registers. These steps are referred as micro-operations. The prefix micro refers to the fact that each of the step is very simple and accomplishes very little.



Execution of a program consists of sequential execution of instructions. Each instruction is executed during an instruction cycle made up of shorter sub-cycles (example - fetch, indirect, execute, interrupt). The performance of each sub-cycle involves one or more shorter operations, that is micro-operations.

(2) Functional Requirements with respect to CPU control Unit :-

The functional requirements of control unit are those functions that the

control unit must perform. And these are the basis for the design and implementation of the control unit. A three step process that lead to the characterization of the control Unit:

- Define the basic elements of the processor.
- Describe the micro-operations that the processor performs.
- Determine the functions that the control unit must perform to cause the micro-operations to be performed.

1) Basic Elements of Processor :- The following are the basic functional elements of a CPU:

ALU: is the functional essence of the computer.

Registers: are used to store data internal to the CPU.

2) Types of Micro-operation :- These operations consist of a sequence of micro operations. All micro instructions fall into one of the following categories:

- Transfer data between registers
- Transfer data from register to external
- Transfer data from external to register
- Perform arithmetic or logical operations

3) Control Functions of Control Unit :- Now we define more explicitly the function of control unit. The control

unit performs two tasks.

- Sequencing: The control unit causes the CPU to step through a series of micro-operations in proper sequence based on the program being executed.
- Execution: The control unit causes each micro-operation to be performed.

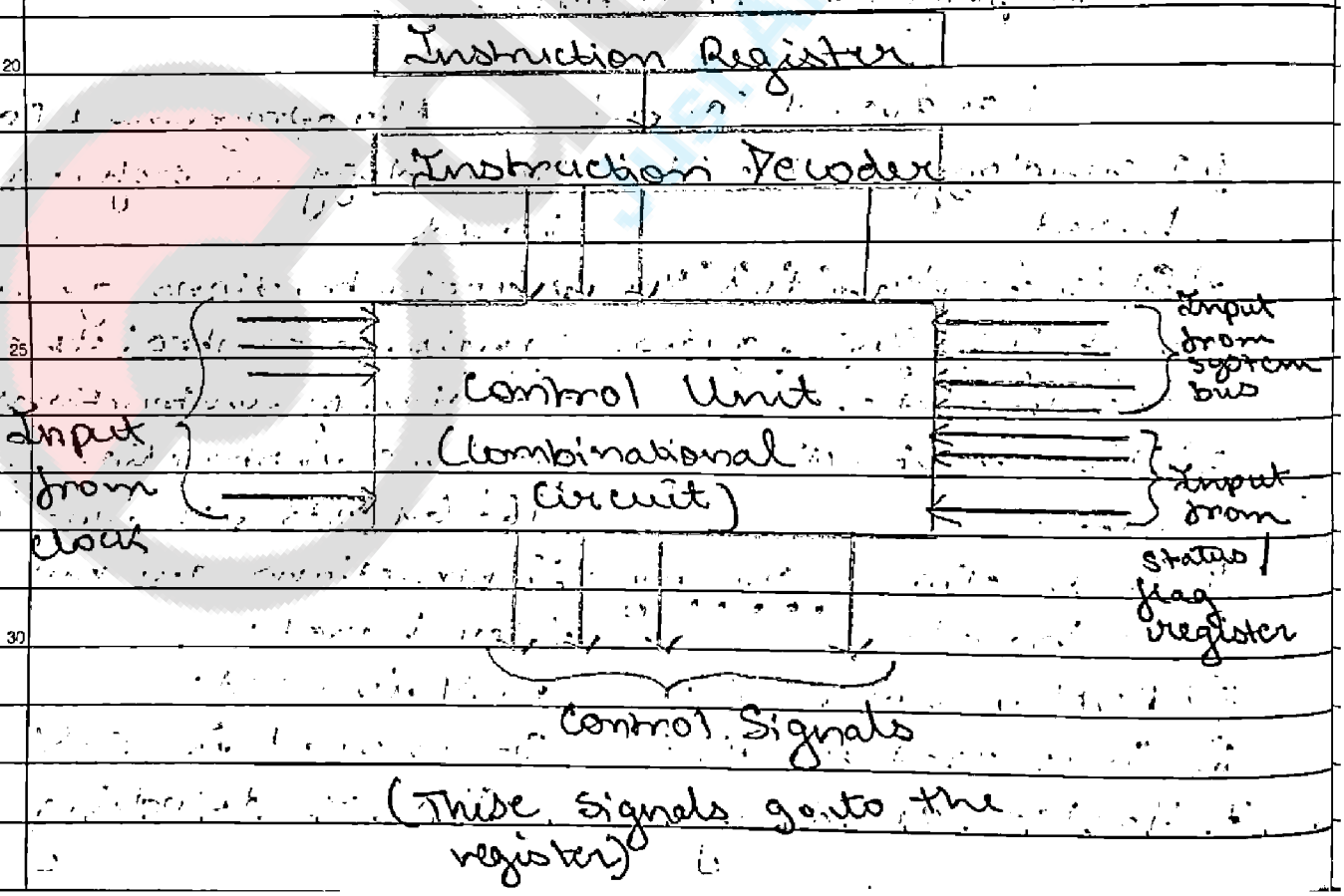
For the control unit to perform its function, it must have inputs that allow it to determine the state of the system and outputs that allow it to control the behavior of the system. These are the external specifications of the control unit. Internally, the control unit must have the logic required to perform sequencing and execution functions.

(3) Difference between Hardwired Control and Microprogrammed Control :-

Hardwired Control	Microprogrammed Control
1) Technology is circuit based	Technology is software based
2) It is implemented through flip-flops, gates, decoders etc.	Microinstructions generate signals to control the execution of instructions.
3) Fixed instruction format.	Variable instruction format (16-64 bits per instruction)
4) Instructions are register based.	Instructions are not register based.
5) ROM is not used.	ROM is used.
6) It is used in RISC	It is used in CISC
7) Faster decoding	Slower decoding

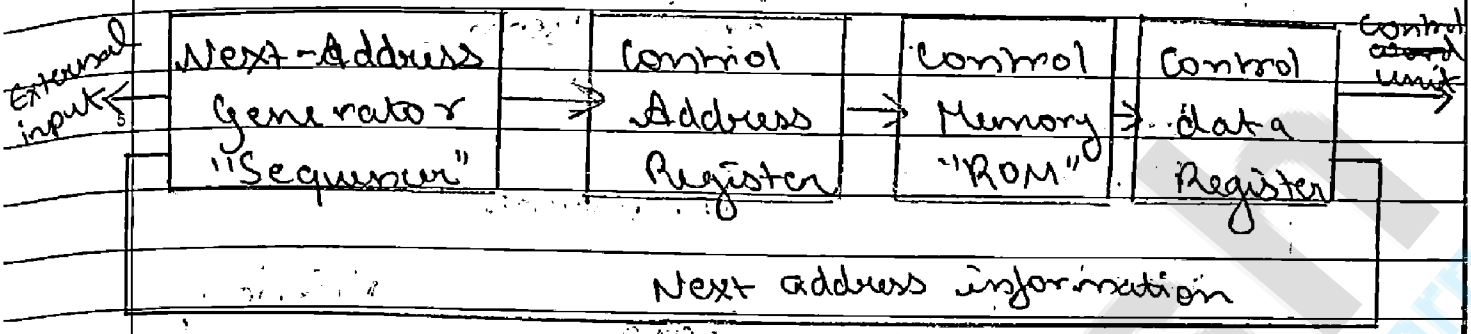
a) Difficult to modify	Easily modified
b) Chip area is less	Chip area is large
c) Speed of operations is fast	Speed of operations is slow because it requires frequent memory accesses
d) To do modifications, the entire unit should be redesigned	Modifications can be implemented by changing the microinstructions in the control memory
e) More costly to implement	Less costly to implement
f) It is difficult to handle complex instructions	It is easier to handle complex instructions
g) There is no control memory usage	Uses control memory

Hardwired Control Organization



Camlin | Page
Date / /

Microprogrammed Control Organization.



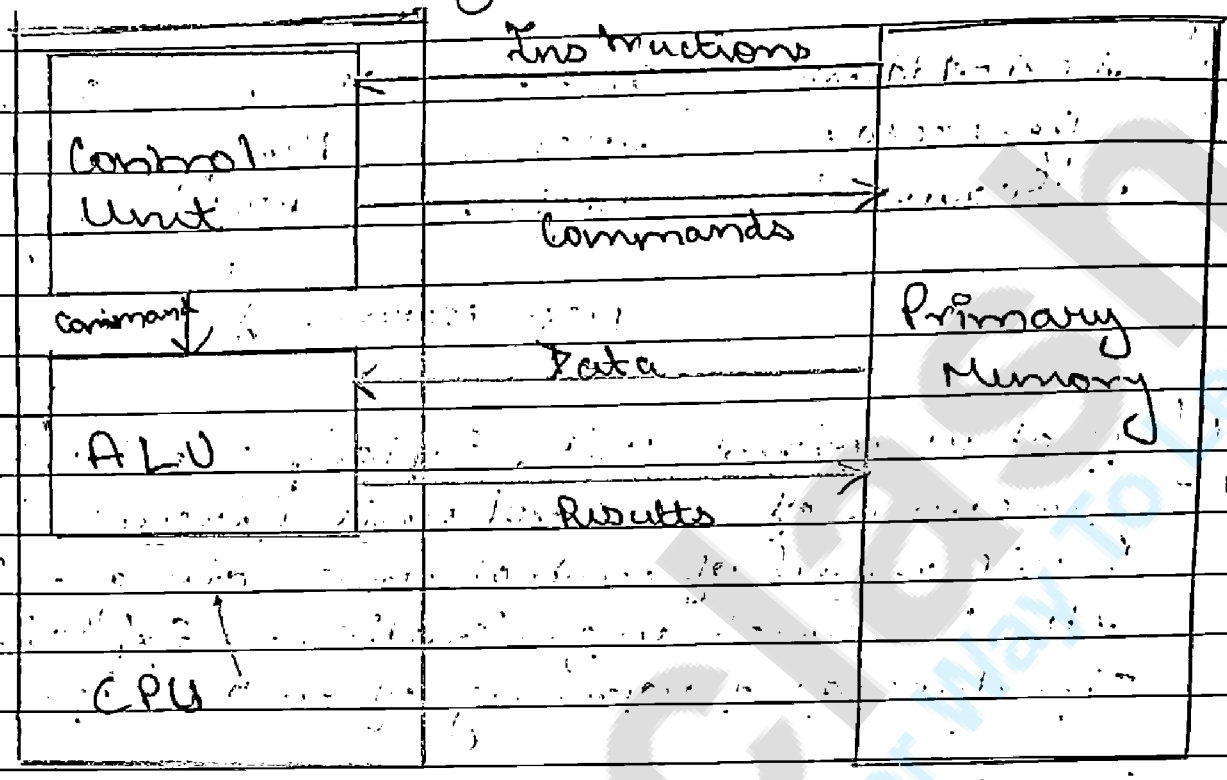
Q] What is control unit? Explain the basic functions of a control unit. Discuss the basic model of control unit along with its internal organization. / Explain the structure and working of control unit.

Control unit co-ordinates the transfer of data between registers of CPU or microprocessor and ALU. Control unit serves the instructions for ALU. Along with this, control unit, as its name implies, controls every other parts of the machine, their co-ordinations, traffic etc.

Thus control unit controls the complete workflow of the CPU. But control unit doesn't take inputs, gives outputs, process data or store data itself; what control unit do is, it controls these operations when they are performed by respective devices.

The purpose of control unit is to run the whole computer. And control unit is ran by the instructions stored in RAM and ROM. So, control unit receives instructions which are stored in RAM and ROM and

controls operations of other connected units or devices through those instructions.



Functions of Control Unit

Functions of Control Unit:- Functions of control unit can be categorized into following 5 categories.

1) Fetching instructions one by one from primary memory and gathering required data and operands to perform those instructions.
2) Sending instructions to ALU to perform additions, multiplication etc.
3) Receiving and sending results of operations of ALU to primary memory.
4) Fetching programs from input and secondary memory and bringing them to primary memory.
5) Sending results from ALU stored in primary memory to output.

Camlin Page
Date / /

Control Units are designed in two ways:-
1) Hardwired control 2) Microprogram control
[write the previous answer]

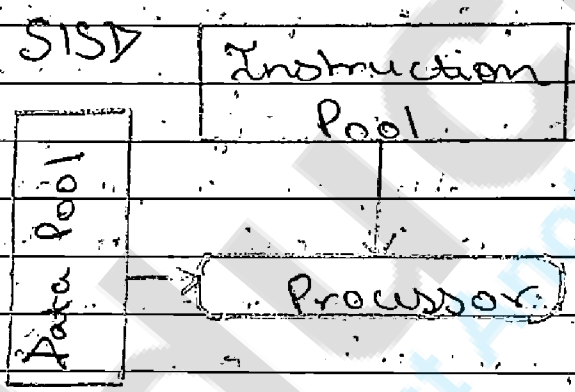
Unit 6

Q1) Flynn's Taxonomy:- Parallel computing is a computing where the jobs are broken into discrete parts that can be executed concurrently. Each part is further broken down to a series of instructions. Instructions from each part execute simultaneously on different CPUs. Parallel systems deal with the simultaneous use of multiple computer resources that can include a single computer with multiple processors, a number of computers connected by a network to form a parallel processing cluster or a combination of both.
Parallel systems are more difficult to program than computers with a single processor because the architecture of parallel computers varies accordingly and the processes of multiple CPUs must be coordinated and synchronized.
The crux of parallel processing are CPUs. Because Based on the number of instruction and data streams that can be processed simultaneously, computing systems are classified into four major categories.

Data Streams \ Instruction Streams	one	many
one	SISD	MISD
many	SIMD	MIMD

Flynn's classification:-

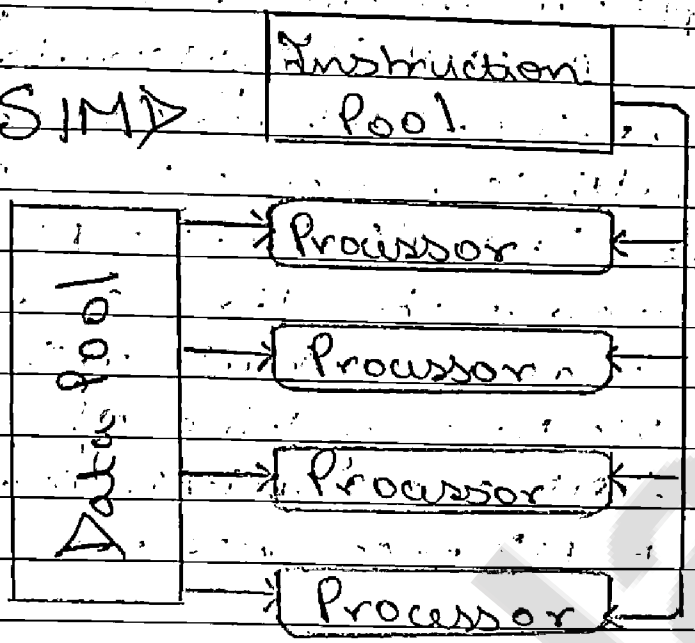
1) Single-instruction, single-data (SISD) system:
An SISD computing system is a uniprocessor machine which is capable of executing a single instruction, operating on a single data stream. In SISD, machine instructions are processed in a sequential manner and computers adopting this model are popularly called sequential computers. Most conventional computers have SISD architecture. All the instructions and data to be processed have to be stored in primary memory.



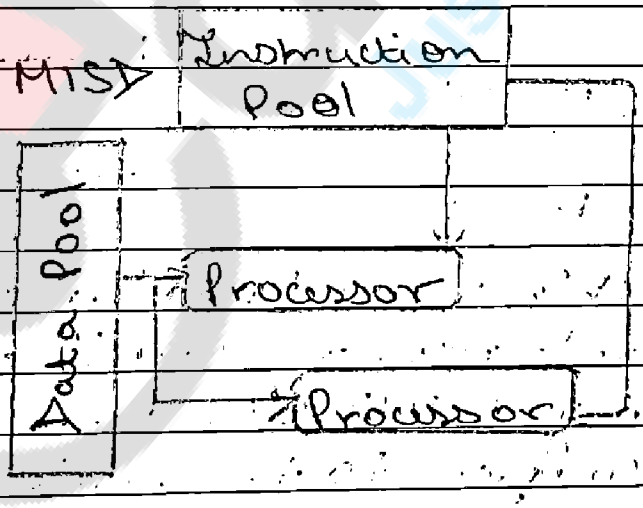
The speed of the processing element in the SISD model is limited (dependent) by the rate at which the computer can transfer information internally. Dominant representative SISD systems are IBM PC, workstations.

2) Single-instruction, multiple-data (SIMD) system: An SIMD system is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams. Machines based on an SIMD model are

will suited to scientific computing since they involve lots of vector and matrix operations.



3) Multiple-instruction, single-data (MISD) systems :- An MISD computing system is a multiprocessor machine capable of executing different instructions on different processing elements but all of them operating on the same dataset.

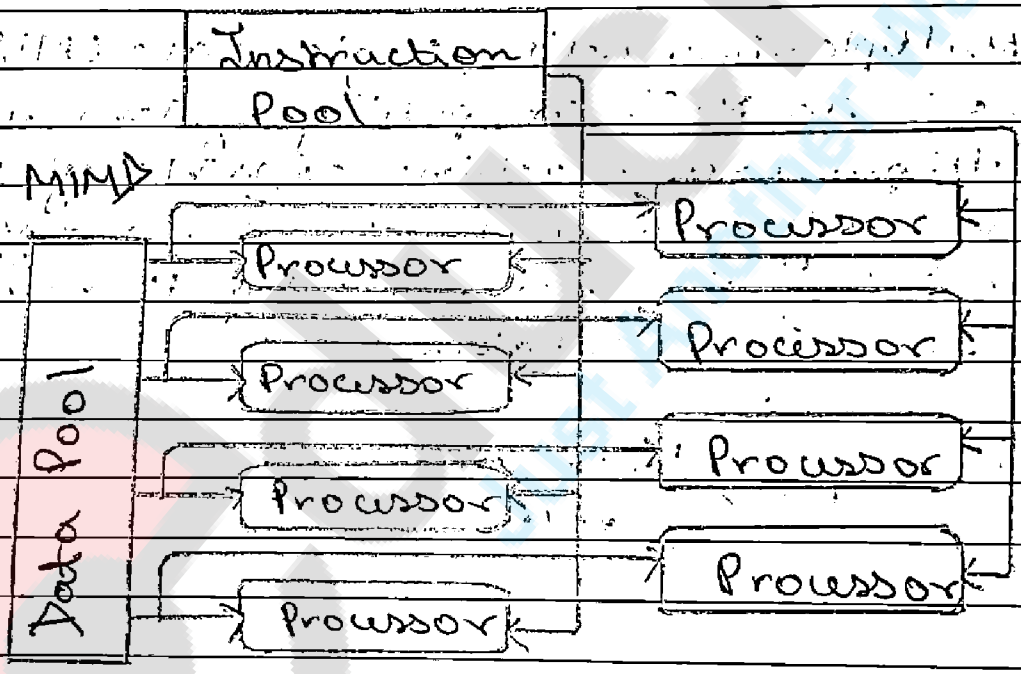


The systems performs different operation on the same data set. Machines built using the MISD model are not useful in most of the application, a few machines are

built, but none of them are available commercially.

4) Multiple-instruction, multiple-data (MIMD) Systems:- An MIMD system is a multiprocessor machine which is capable of executing multiple instructions on multiple data sets. Each PE in the MIMD model has separate instruction and data streams; therefore machines built using this model are capable to any kind of application. Unlike SIMD and ~~MISD~~ MISD machines, PEs in MIMD machines work asynchronously.

PE = Processing Element



MIMD machines are broadly categorized into shared-memory MIMD and distributed-memory MIMD based on the way PEs are coupled to the main memory. In the shared memory MIMD model (tightly coupled multiprocessor systems) all the PEs are connected to a single global memory and they all have

access to it. The communication between PEs in this model takes place through the shared memory, modification of the data stored in the global memory by one PE is visible to all other PEs. Dominant representative shared memory MIMD systems are Silicon Graphics machines and Sun/IBM's SMP (Symmetric Multi-Processing).

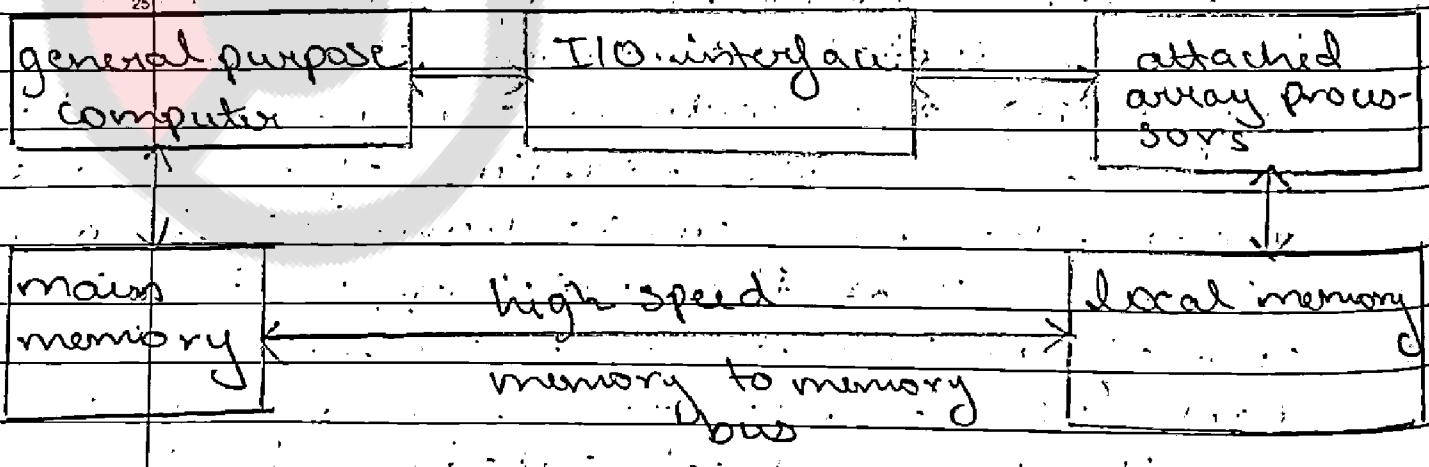
In Distributed memory MIMD machines (loosely coupled multiprocessor systems) all PEs have a local memory. The communication between PEs in this model takes place through the interconnection network (the inter process communication channel or IPC). The network connecting PEs can be configured to tree, mesh or in accordance with the requirement. The shared-memory MIMD architecture is easier to program but is less tolerant to failures and harder to extend with respect to the distributed memory MIMD model. failures in a shared-memory MIMD affect the entire system, whereas this is not the case of the distributed model, in which each of the PEs can be easily isolated. Moreover, shared memory MIMD architectures are less likely to scale because the addition of more PEs leads to memory contention. This is a situation that does not happen in the case of distributed memory, in which each PE has its own memory. As a result of practical outcomes and

user's requirement, distributed memory MIMD architecture is superior to the other existing models.

Q2) Array Processors :- Array processors are also known as multiprocessors or vector processors. They perform computations on large arrays of data. Thus, they are used to improve the performance of the computer.

Types of Array Processors :- There are basically two types of array processors:
1) Attached Array Processors
2) SIMD Array Processors

1) Attached Array Processors :- An attached array processor is a processor which is attached to a general purpose computer and its purpose is to enhance and improve the performance of that computer in numerical computational tasks. It achieves high performance by means of parallel processing with multiple functional units.



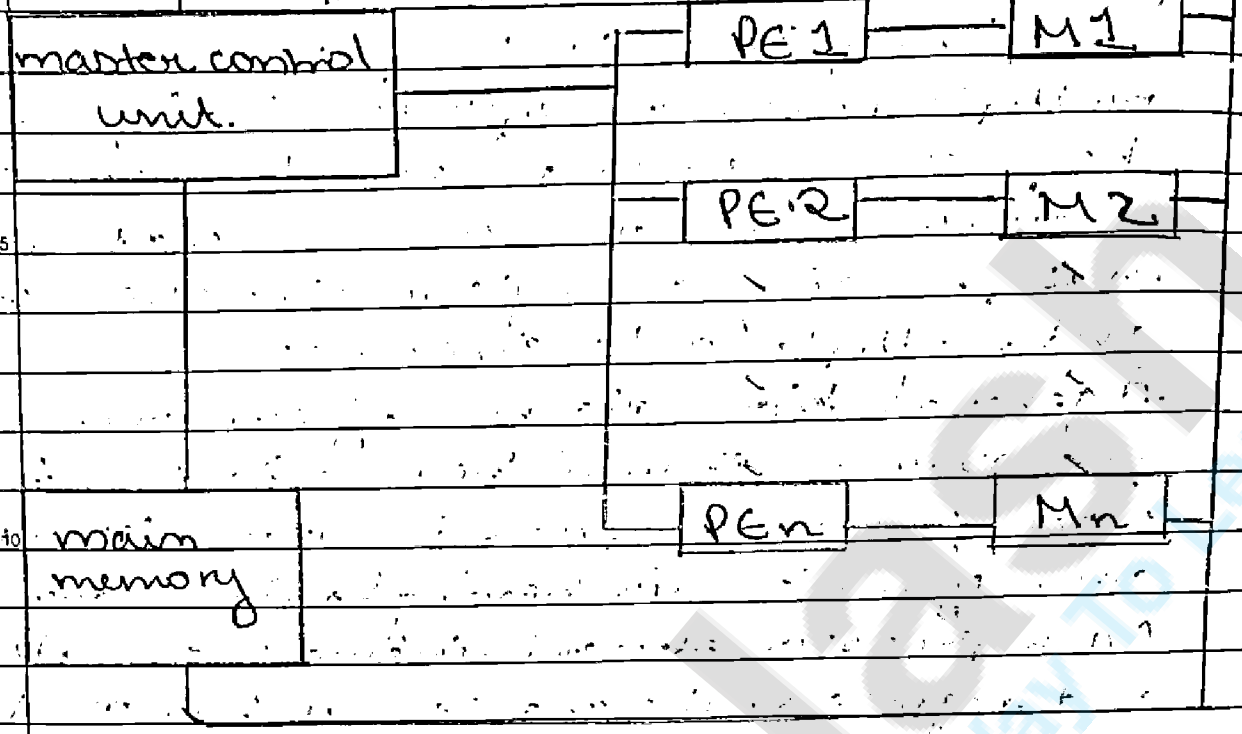
Camlin Page
Date / /

2) SIMD Array Processors :- SIMD is the organization of a single computer containing multiple processors operating in parallel. The processing units are made to operate under the control of a common control unit, thus providing a single instruction stream and multiple data streams.

A general block diagram of an array processor is shown below. It contains a set of identical processing elements (PE's), each of which is having a local memory M. Each processor element includes an ALU and registers. The master control unit controls all the operations of the processor elements. It also decodes the instructions and determines how the instruction is to be executed.

The main memory is used for storing the program. The control unit is responsible for fetching the instructions. Vector instructions are send to all PE's simultaneously and results are returned to the memory.

SIMD processors are highly specialized computers. They are only suitable for numerical problems that can be expressed in vector or matrix form and they are not suitable for other types of computations.



Why use Array Processor?

1) Array processors increases the overall instruction processing speed.
2) As most of the Array processors operates asynchronously from the host CPU, hence it improves the overall capacity of the system.
3) Array processors has its own local memory, hence providing extra memory for systems with low memory.

Q3) Cluster:- A computer cluster is a group of tightly coupled computers that work together closely so that it can be viewed as a single computer. Clusters are commonly connected through fast local area network (LAN). Clusters have evolved to support application ranging from ecommerce, to high performance database applications.

Camlin Page
Date / /

Clusters are usually deployed to improve speed and reliability over that provided by a single computer, while typically being much more cost effective than single computer of comparable speed or reliability.
In cluster computers each node within a cluster is an independent system with its own file system. Because processors on 1 node cannot directly access the memory on the other nodes, programs or software running on clusters usually employ a procedure called "message passing" to get data and execution code from one node to another.
Clusters computing can also be used as a relatively low-cost form of parallel processing for scientific and other applications that lend themselves on parallel operations.

Architecture of cluster :-

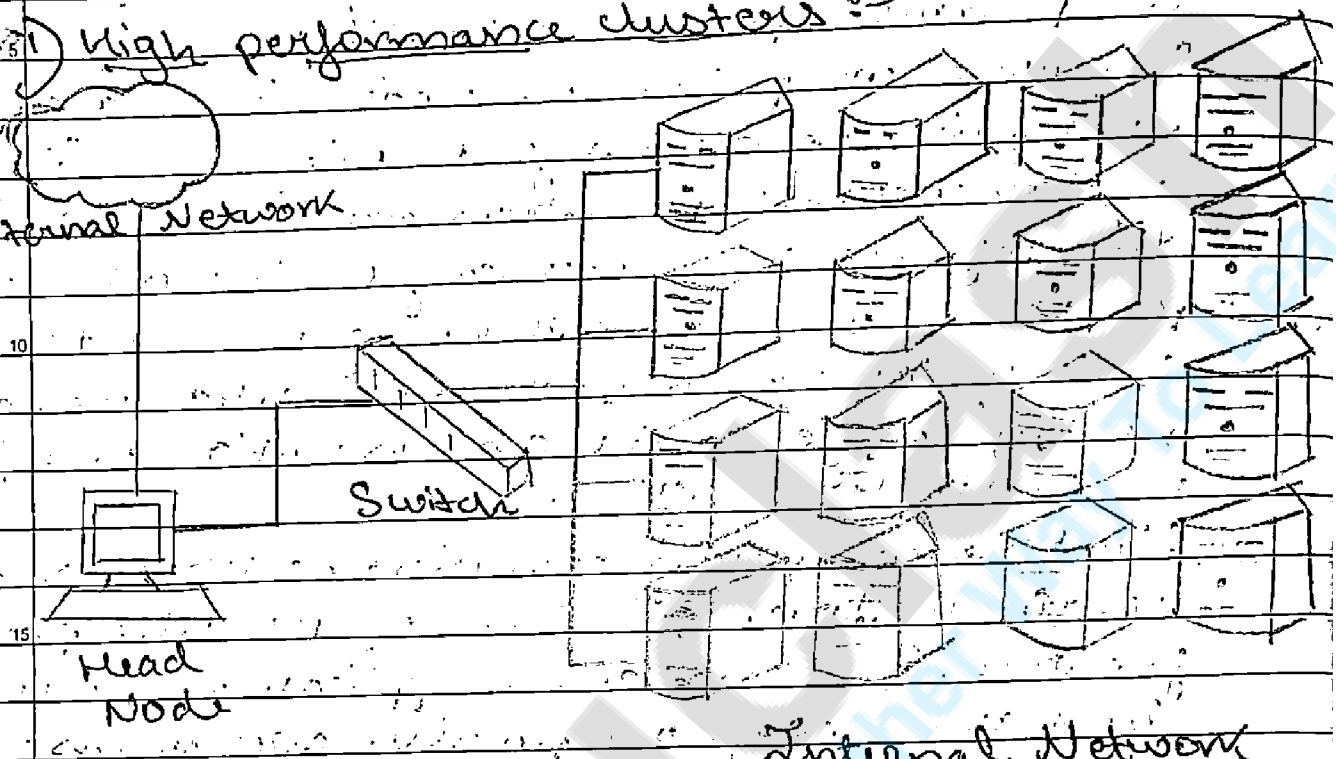
A cluster is a type of parallel / distributed processing system which consists of a collection of interconnected stand-alone computers co-operatively working together as a single, integrated computing resource.

A node :- A single or multiprocessor system with memory, I/O facilities & OS.
- Generally two or more computers (nodes) are connected together in a single cabinet or physically operated & connected via LAN.
- Appears as a single system to users and applications.
- Provide a cost effective way to gain features and benefits.

Types of clusters :-

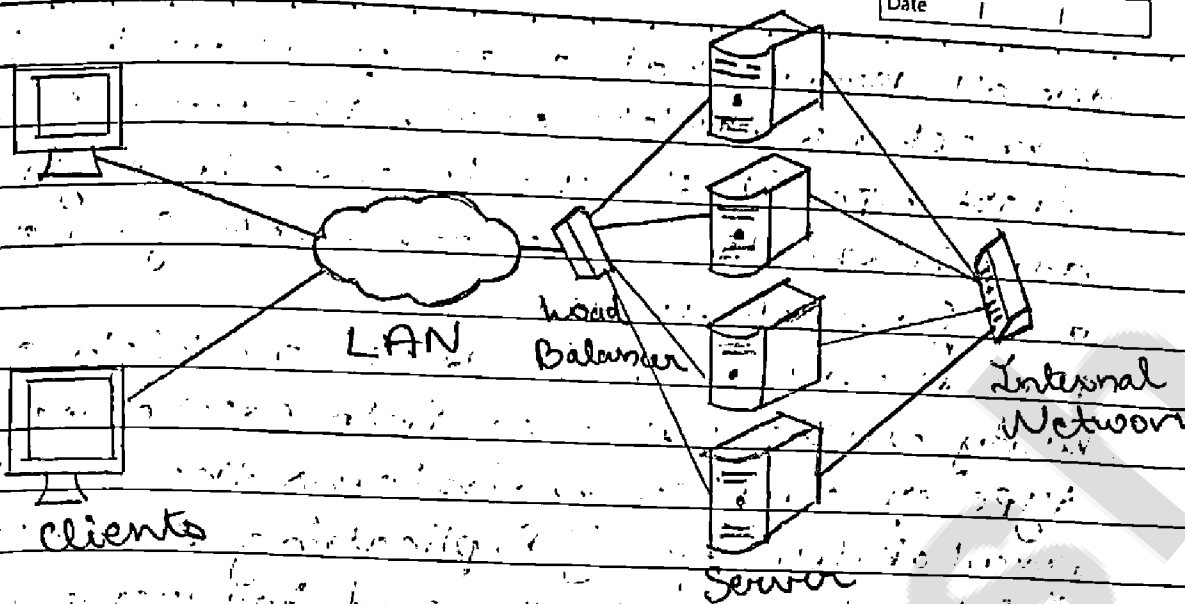
1) High performance (HP) clusters
2) Load balancing cluster
3) High availability cluster

1) High performance clusters :-



As the name suggests, the objective behind forming high performance clusters is to deliver high performance computer systems. They run parallel programs that are required for time-exhaustive computations. Such a kind of clusters are commonly preferred by scientific industries. The basic aim of high performance cluster is thus increasing performance by intelligently sharing the work load. That is why they are used by life-sciences research industry, graphics rendering sciences etc.

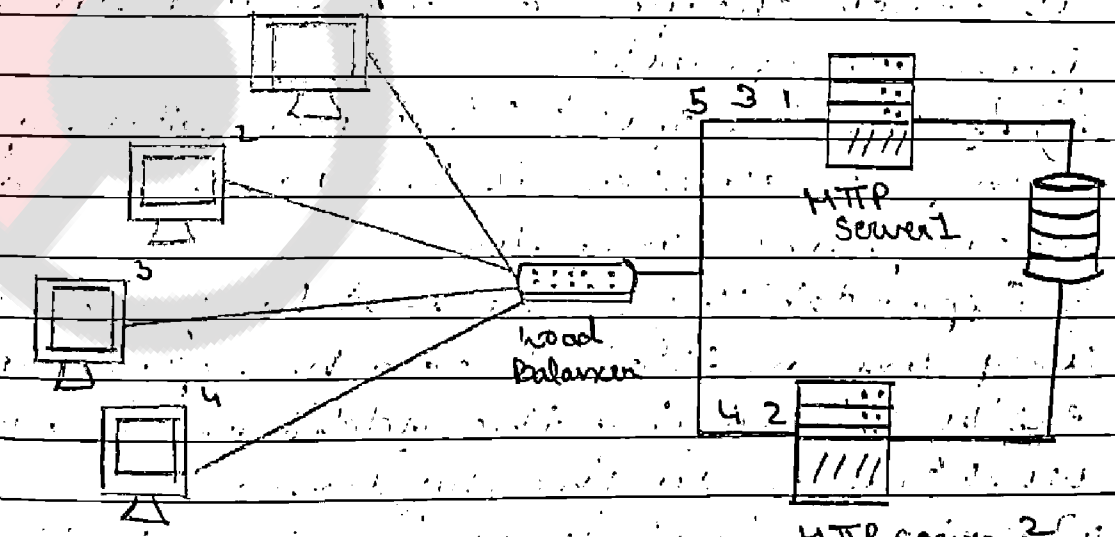
2) Load Balancing Clusters :-



Such types of clusters work on the idea of distributing loads among different servers. They can aim to provide increased network capacity, eventually increasing the performance. E-commerce businesses and internet service providers use load balancing clusters to manage user requests and deliver high performance.

Despite so much distribution at the back end into the user it appears to be a single system only.

3) High Availability cluster :-



We all know that a computer is basically a machine which can stop working at least expected times. So, how do system administrators manage such failovers and fix such issues? Here the role of failover or high availability clusters comes into existence. Such type of cluster maintains the service availability by a) Replicating servers b) Redundant software and hardware configuration

So, every system is monitoring the other and takes up request if any one node fails. These types of clusters are beneficial for those users who have a high-dependency on their computer systems. Eg include the e-commerce businesses, news channels, websites, databases etc.

Advantages of using cluster computing:-

1) Cost efficiency :- In a cluster computing cost efficiency is the ratio of cost to output, that is the connecting group of the computer as ~~to~~ computer cluster is much cheaper as compared to mainframe computers.

2) Processing speed :- The processing speed of computer cluster is the same as a mainframe computer.

3) Expandability :- The best benefit of cluster computing is that it can be expanded easily by adding the additional desktop workstation to the system.

4) High availability of resources :- If any

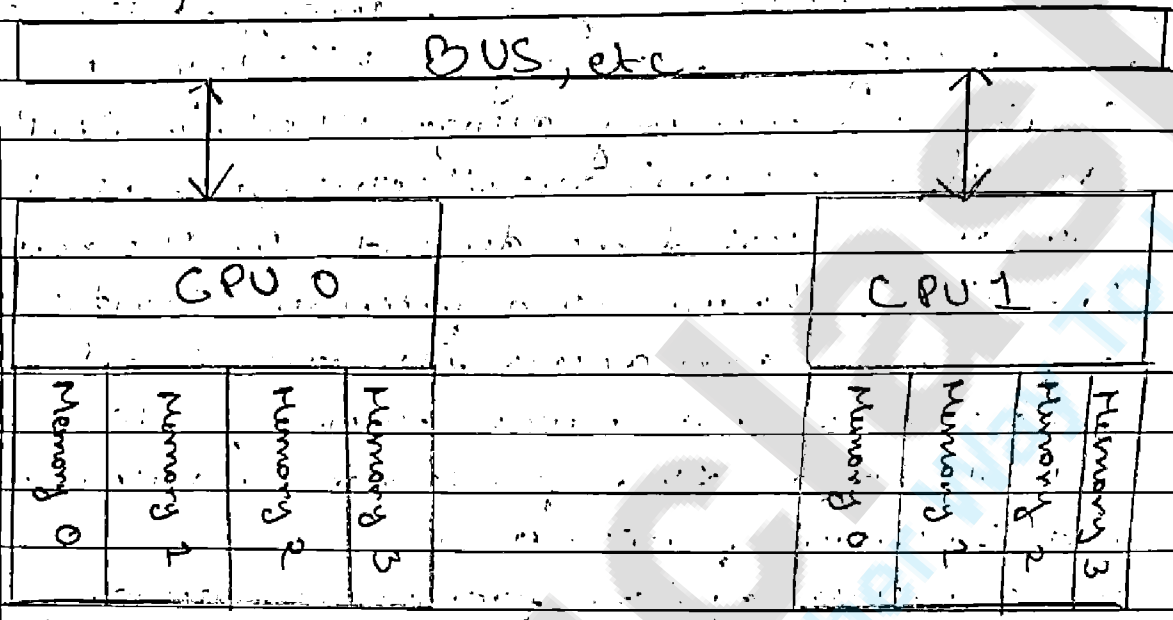
node fails in a computer cluster, another node within the cluster continue to provide uninterrupted processing. When a mainframe system fails, the entire fails.

Q4) NUMA :- Non-Uniform Memory Access or Non-Uniform Memory Architecture (NUMA) is a physical memory design used in SMP (Symmetric multiprocessors) architecture, where the memory access time depends on the memory location relative to a processor. Under NUMA, a processor can access its own local memory faster than non-local memory, i.e, memory local to another processor or memory shared between processors.

On a NUMA system, processors, memory and I/O are grouped together into cells. The latency and bandwidth characteristics of communication within a cell are "fast" while going outside a cell is "slow". Since the memory in NUMA systems is physically distributed but logically shared, these systems offer better performance to applications that are optimized to use their features. For non-optimized applications, they still offer better performance since the default behavior is designed to be benign - if not beneficial - and they still have access to much larger shared resources of memory, CPU's and disk space.

NUMA Architecture :- In NUMA architecture, each processor has a 'local' bank of memory, to which it has much closer

(lower latency) access. The whole system may still operate as a one unit, and all memory is basically accessible from everywhere but at a potentially higher latency and lower performance.



Fundamentally, some memory locations ('local' ones) are faster, that is, cost less to access, than other locations ('remote' ones attached to other processors).

Q5] Multiprocessor Systems: Structure & Interconnection Networks:-

An interconnection network in a parallel machine transfers information from any source node to any desired destination node. This task should be completed with as small latency as possible. It should allow a large number of such transfers to take place concurrently. Moreover, it should be inexpensive as compared to the cost of the rest of the machine.

The network is composed of link and switches, which helps to send the information from the source node to the destination node. A network is specified by its topology, routing algorithm, switching strategy and flow control mechanism.

Organizational Structure :- Interconnection networks are composed of following three basic components -

1) Links :- A link is a cable of one or more optical fibers or electrical wires with a connector at each end attached to a switch or network interface port. Through this, an analog signal is transmitted from one end, received at the other end to obtain the original digital information stream.

2) Switches :- A switch is composed of a set of input and output ports, an internal "cross-bar" connecting all input to all output, internal buffering and control logic to effect the input-output connection at each point in time. Generally, the number of input ports is equal to the number of output ports.

3) Network Interfaces :- The network interface behaves quite differently than switch nodes and may be connected via special links. The network interface formats the packets and constructs the routing and control information. It may have input and output buffering, compared to a switch. It may perform end-to-end error checking and flow control. Hence, its.

cost is influenced by its processing complexity, storage capacity, and number of ports.

Interconnection Network :- Interconnection networks are composed of switching elements. Topology is the pattern to connect the individual switches to other elements, like processors, memories and other switches. A network allows exchange of data between processors in the parallel system.

1) Direct connection networks :- Direct networks have point-to-point connections between neighboring nodes. These networks are static, which means that the point-to-point connections are fixed. Some examples of direct networks are rings, meshes and cubes.

2) Indirect connection networks — Indirect networks have no fixed neighbours. The communication topology can be changed dynamically based on the application demands. Indirect networks can be subdivided into three parts : bus networks, multistage networks and crossbar switches.

- Bus networks :- A bus network is composed of a number of bit lines onto which a number of resources are attached. When busses use the same physical lines for data and addresses, the data and the address lines are time multiplexed. When there are multiple bus-masters

attached to the bus, an arbiter is required

• Multistage networks - A multistage network consists of multiple stages of switches. It is composed of 'axb' switches which are connected using a particular interstage connection pattern. Small 2x2 switch elements are a common choice for many multistage networks. The number of stages determine the delay of the network. By choosing different interstage connection patterns, various types of multistage network can be created.

• Crossbar switches - A crossbar switch contains a matrix of simple switch elements that can switch on and off to create or break a connection. Turning on a switch element in the matrix, a connection between a processor and a memory can be made. Crossbar switches are non-blocking, that is all communication permutations can be performed without blocking.

Q6] Multicore Computer Organization :-
A multi-core processor is one which combines two or more independent processors into a single package, often a single integrated circuit. Examples are Intel Core i7, intel core 2 duo.

Problems with Single Core :- a) To execute the tasks faster you must increase the clock time.

b) Increasing clock times too high drastically

increases power consumption and heat dissipation to extremely high levels, making the processor inefficient.

Multi Core Solution :- a) Creating two cores or more on the same die increases processing power while keeping clock speeds at an efficient level.

b) A processor with 2 cores running at efficient clock speeds can process instructions with similar speed to a single core processor running at twice the clock speed, yet the dual core processor would still consume less energy.

c) Better performance :- for the multi tasking eg :- Burning CD with graphic works at the same time.

d) lower consumption and heat generation caused from the advance of CPU clock speed.

e) Save the room of motherboard :- Two single cores → On one die - We can use this room more efficiently

f) Simplicity - We need additional systems to control the several single cores.

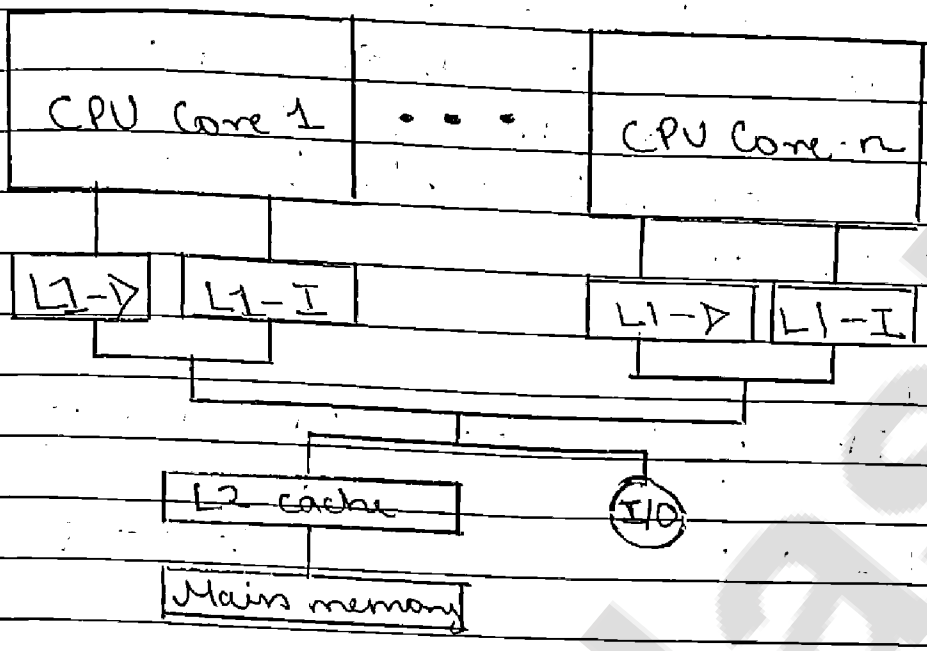
g) Economical efficiency - A dual core is much cheaper than two single cores.

Multicore Organization :-

The main variables in a multicore organization are as follows :-

1) The number of core processors on the chip
2) The number of levels of cache memory
3) The amount of cache memory that is shared

Multicore Organization

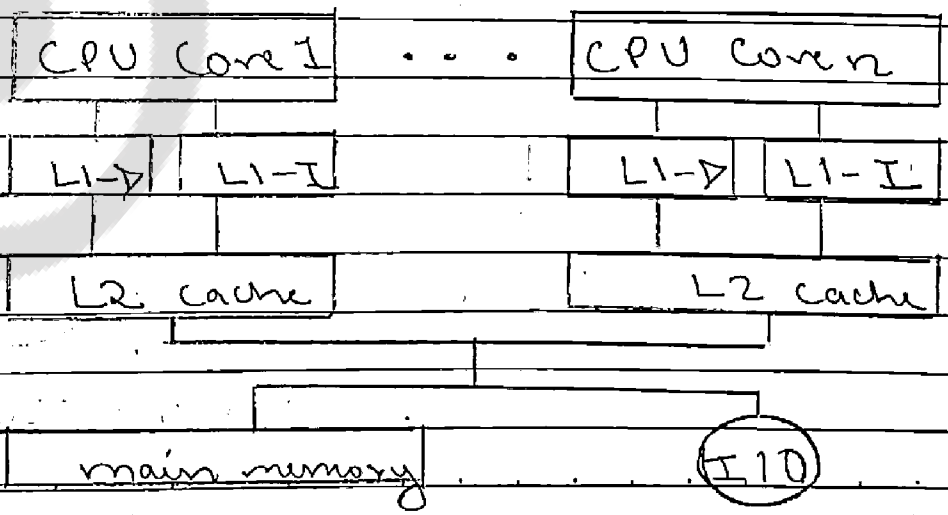


Dedicated L1 cache

Figure shows an organization found in some of the earlier multicore computer chips and is still seen in embedded chips. In this organization, the only on-chip cache is L1 cache, each core having its own dedicated L1 cache.

Almost invariably, the L1 cache is divided into instruction and data caches.

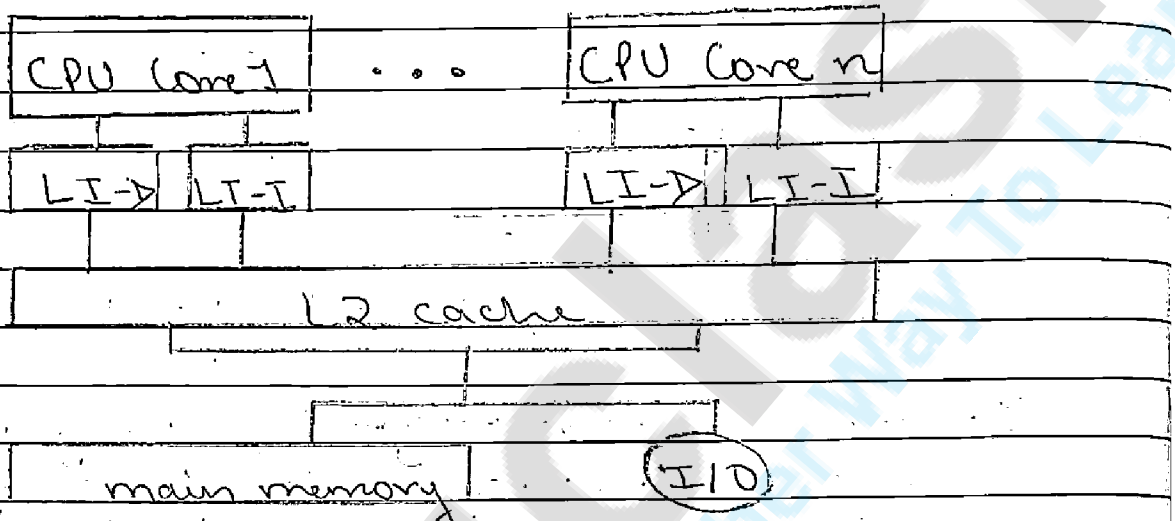
An example of this organization is the ARM11 MPCore.



Dedicated L2 cache

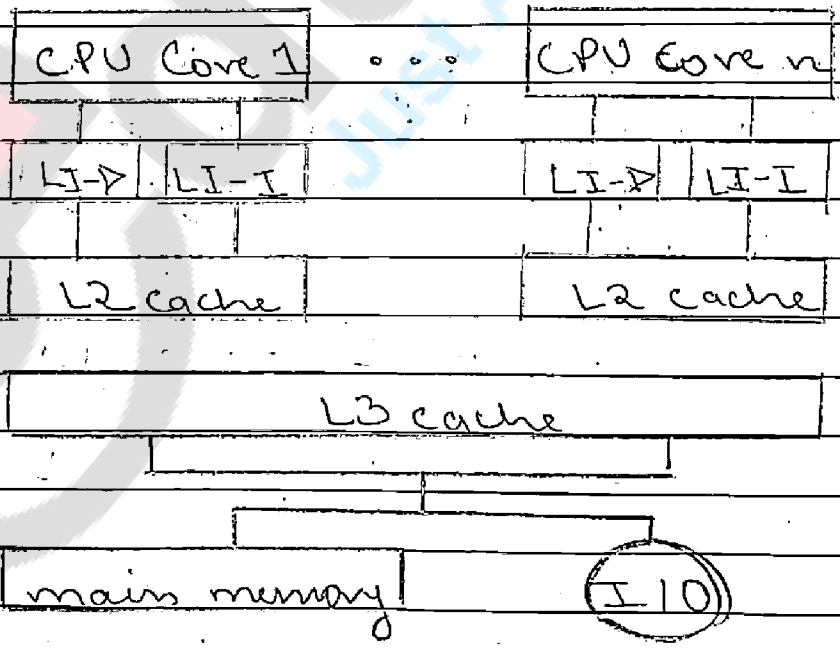
The organization is also one in which there is no on-chip cache sharing.
In this, there is enough area available on the chip to allow for L2 cache.
An example of this organization is the AMD Opteron

③



Shared L2 cache

④



Shared L3 cache

Camlin Page
Date / /

③ figure shows allocation of chip space to memory, but with the use of a shared L2 cache.
The Intel Core Due has this organization.
The amount of cache memory available on the chip continues to grow, performance considera-
④ tions dictate splitting off a separate L3 cache, with dedicated L1 and L2 caches for each core processor.
The intel Core I7 is an example of this Orga-
nization.