Total No. of Questions : 8] *[Total No. of Printed Pages : 4*

Roll No ....

IT-402 (CBGS)

B.Tech., IV Semester

Examination, November 2019

Choice Based Grading System (CBGS)

Computer Architecture

Time : Three Hours

Maximum Marks : 70

Note: i) Attempt any five questions.
किन्हीं पाँच प्रश्नों को हल कीजिए।

ii) All questions carry equal marks.
सभी प्रश्नों के समान अंक हैं।

iii) In case of any doubt or dispute the English version question should be treated as final.
किसी भी प्रकार के संदेह अथवा विवाद की स्थिति में अंग्रेजी भाषा के प्रश्न को अंतिम माना जायेगा।

1. a) Explain computer architecture and organization. 7
कम्प्यूटर आर्किटेक्चर (वास्तुकला) और उसके संगठन की व्याख्या करें।

b) Explain the eight great ideas invented by computer architecture. 7
कम्प्यूटर आर्किटेक्चर (वास्तुकला) द्वारा आविष्कार किये गये 8 महान विचारों की व्याख्या करें।

PTO

2. a) What is data hazard? How do you overcome it? And discuss its side effects. 7

डाटा जोखिम (हजार्ड) क्या है? आप इसे कैसे दूर करेंगे? और इसके दुष्प्रभावों की चर्चा करे।

b) List the advantages of multibus organization? 7

मल्टीबस संगठन के फायदे सूचीबद्ध करें।

3. a) What is an addressing mode? Explain the various addressing modes with suitable examples. 7

एड्रेसिंग मोड क्या है? उपयुक्त उदाहरण के साथ विभिन्न एड्रेसिंग मोड बताइये।

b) Draw the typical block diagram of a DMA controller and explain how it is used for direct data transfer between memory and peripherals? 7

एक डी एम ए नियंत्रक के विशिष्ट ब्लॉक आरेख को बनाएँ और यह बताएँ की इसका उपयोग स्मृति और बाह्य उपकरणों के मध्य सीधे डाटा हस्तांतरण के लिए कैसे किया जाता है।

4. a) What is virtual memory? Explain in detail about how virtual memory is implemented with neat diagram? 7

वर्चुअल मेमोरी क्या है? वर्चुअल आरेख को स्वच्छ आरेख के साथ कैसे कार्यान्वित किया जाता है, इसे विस्तार से समझाइए।

b) What is memory hierarchy? What is the need to implement memory as a hierarchy? 7

मेमोरी पदानुक्रम क्या है? स्मृति को पदानुक्रम के रूप में लागू करने की आवश्यकता क्या है?

5. a) Explain how the instruction pipeline works. 7
निर्देश पाइपलाइन कैसे काम करता है, विस्तार से समझाइये।

b) What are the various situations where an instruction pipeline can stall? Illustrate with an examples. 7
विभिन्न परिस्थितियाँ क्या है, जहाँ एक निर्देश पाइपलाइन स्टाल कर सकती है। उदाहरण सहित समझाइये।

6. a) Explain computer main memory? And difference between main memory and cache memory. 7
कम्प्यूटर के मुख्य मेमोरी की व्याख्या करे? और मुख्य मेमोरी और कैश मेमोरी के मध्य अन्तर स्पष्ट करें।

b) Describe the role of system software to improve the performance of a computer. 7
कम्प्यूटर के प्रदर्शन को बेहतर बनाने के लिए सिस्टम सॉफ्टवेयर की भूमिका का वर्णन करें।

7. a) What are the disadvantages of increasing the number of stages in pipelined processing? 7
पाइपलाइन प्रसंस्करण में चरणों की संख्या बढाने के क्या नुकसान है?

b) Describe the USB architecture with the help of a neat diagram. 7
एक साफ आरेख द्वारा यू एस बी आर्किटेक्चर का वर्णन करें।

8. Answer the following. 14
i) Explain IOP
ii) Explain Cache memory
iii) Write a short notes Bus and memory Transfers
iv) Compare UMA and NUMA multi processors.

निम्नलिखित के उत्तर दीजिए।

i) IOP समझाइये।

ii) कैश मेमोरी की व्याख्या करे

iii) संक्षिप्त नोट लिखिये बस और मेमोरी ट्रांसफर

iv) UMA और NUMA मल्टीप्रोसेसर की तुलना करे

******

Q.1	(a)	**Computer Architecture:** Computer Architecture is a functional description of requirements and design implementation for the various parts of computer.It deals with functional behavior of computer system. It comes before the computer organization while designing a computer. **Computer Organization:** Computer Organization comes after the decide of Computer Architecture first. Computer Organization is how operational attribute are linked together and contribute to realise the architectural specification. Computer Organization deals with structural relationship.

1.	Architecture describes what the computer does.	Organization describes how it does it.
2.	Computer Architecture deals with functional behavior of computer system.	Computer Organization deals with structural relationship.
3.	In above figure, its clear that it deals with high-level design issue.	In above figure, its also clear that it deals with low-level design issue.
4.	Architecture indicates its hardware.	Where, Organization indicates its performance.
5.	For designing a computer, its architecture is fixed first.	For designing a computer, organization is decided after its architecture.
6.	Computer Architecture is also called as instruction set architecture.	Computer Organization is frequently called as micro architecture.
7.	Computer Architecture comprises logical functions such as instruction sets, registers, data types and addressing modes.	Computer Organization consists of physical units like circuit designs, peripherals and adders.

8.	Architecture coordinates between the hardware and software of the system.	Computer Organization handles the segments of the network in a system.

(b) Application of the following great ideas has accounted for much of the tremendous growth in computing capabilities over the past 50 years.

1. Design for Moore’s Law

   The one constant for computer designers is rapid change, which is driven largely by Moore’s Law. It states that integrated circuit resources double every 18–24 months. Moore’s Law resulted from a 1965 prediction of such growth in IC capacity made by Gordon Moore, one of the founders of Intel. As computer designs can take years, the resources available per chip can easily double or quadruple between the start and finish of the project. Like a skeet shooter, computer architects must anticipate where the technology will be when the design finishes rather than design for where it starts. We use an “up and to the right” Moore’s Law graph to represent designing for rapid change.

2. Use Abstraction to Simplify Design

   Both computer architects and programmers had to invent techniques to make themselves more productive, for otherwise design time would lengthen as dramatically as resources grew by Moore’s Law. A major productivity technique for hardware and soft ware is to use abstractions to represent the design at different levels of representation; lower-level details are hidden to offer a simpler model at higher levels. We’ll use the abstract painting icon to represent this second great idea.

3. Make the common case fast

   Making the common case fast will tend to enhance performance better than optimizing the rare case. Ironically, the common case is often simpler than the rare case and hence is often easier to enhance. This common sense advice implies that you know what the common case is, which is only possible with careful experimentation and measurement. We use a sports car as the icon for making the common case fast, as the most common trip has one or two passengers, and it’s surely easier to make a fast sports car than a fast minivan.

4. Performance via parallelism

   Since the dawn of computing, computer architects have offered designs that get more performance by performing operations in parallel. We’ll see many examples of parallelism in this book. We use multiple jet engines of a plane as our icon for parallel performance.

5. Performance via pipelining

A particular pattern of parallelism is so prevalent in computer architecture that it merits its own name: pipelining. For example, before fire engines, a “bucket brigade” would respond to a fire, which many cowboy movies show in response to a dastardly act by the villain. The townsfolk form a human chain to carry a water source to fire, as they could much more quickly move buckets up the chain instead of individuals running back and forth. Our pipeline icon is a sequence of pipes, with each section representing one stage of the pipeline.

6. Performance via prediction

   Following the saying that it can be better to ask for forgiveness than to ask for permission, the next great idea is prediction. In some cases it can be faster on average to guess and start working rather than wait until you know for sure, assuming that the mechanism to recover from a misprediction is not too expensive and your prediction is relatively accurate. We use the fortune-teller’s crystal ball as our prediction icon.

7. Hierarchy of memories

   Programmers want memory to be fast, large, and cheap, as memory speed often shapes performance, capacity limits the size of problems that can be solved, and the cost of memory today is often the majority of computer cost. Architects have found that they can address these conflicting demands with a hierarchy of memories, with the fastest, smallest, and most expensive memory per bit at the top of the hierarchy and the slowest, largest, and cheapest per bit at the bottom. Caches give the programmer the illusion that main memory is nearly as fast as the top of the hierarchy and nearly as big and cheap as the bottom of the hierarchy. We use a layered triangle icon to represent the memory hierarchy. The shape indicates speed, cost, and size: the closer to the top, the faster and more expensive per bit the memory; the wider the base of the layer, the bigger the memory.

8. Dependability via redundancy

   Computers not only need to be fast; they need to be dependable. Since any physical device can fail, we make systems dependable by including redundant components that can take over when a failure occurs and to help detect failures. We use the tractor-trailer as our icon, since the dual tires on each side of its rear axels allow the truck to continue driving even when one tire fails. (Presumably, the truck driver heads immediately to a repair facility so the fl at tire can be fixed, thereby restoring redundancy!)

Q.2 (a)

Data hazards occur when instructions that exhibit data dependence modify data in different stages of a pipeline. Ignoring potential data hazards can result in race conditions (also termed race hazards). There are three situations in which a data hazard can occur:

1. Read after write (RAW), a *true dependency*
2. Write after read (WAR), an *anti-dependency*
3. Write after write (WAW), an *output dependency*

Consider two instructions i1 and i2, with i1 occurring before i2 in program order.

Read after write (RAW

(i2 tries to read a source before i1 writes to it) A read after write (RAW) data hazard refers to a situation where an instruction refers to a result that has not yet been calculated or retrieved. This can occur because even though an instruction is executed after a prior instruction, the prior instruction has been processed only partly through the pipeline.

Example

For example:

i1. **R2** <- R5 + R3
i2. R4 <- **R2** + R3

The first instruction is calculating a value to be saved in register R2, and the second is going to use this value to compute a result for register R4. However, in a pipeline, when operands are fetched for the 2nd operation, the results from the first have not yet been saved, and hence a data dependency occurs.

A data dependency occurs with instruction i2, as it is dependent on the completion of instruction i1.

Write after read (WAR)

(i2 tries to write a destination before it is read by i1) A write after read (WAR) data hazard represents a problem with concurrent execution.

Example

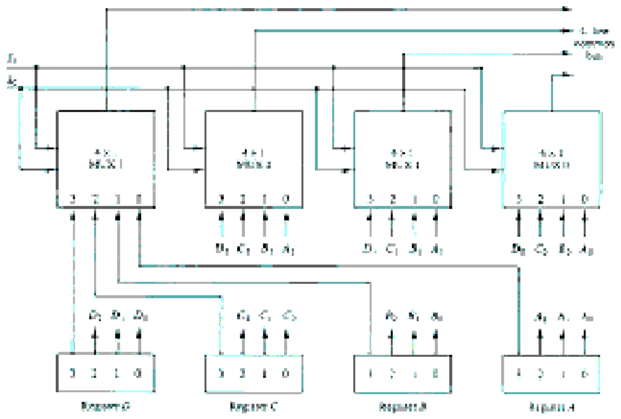
For example:

i1. R4 <- R1 + **R5**
i2. **R5** <- R1 + R2

In any situation with a chance that i2 may finish before i1 (i.e., with concurrent execution), it must be ensured that the result of register R5 is not stored before i1 has had a chance to fetch the operands.

Write after write (WAW)

(i2 tries to write an operand before it is written by i1) A write after write (WAW) data hazard may occur in a concurrent execution environment.

		Example For example: i1. **R2** <- R4 + R7 i2. **R2** <- R1 + R3 The write back (WB) of i2 must be delayed until i1 finishes executing.
	(b)	A typical digital computer has many registers, and paths must be provided to transfer information from one register to another. The number of wires will be excessive if separate lines are used between each register and all other registers in the system. A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time. Control signals determine which register is selected by the bus during each particular register transfer. The multiplexers select the source register whose binary information is then placed on the bus. For example, the construction of a bus system for four registers is shown in Figure Each register has four bits, numbered 0 through 3. The bus consists of four 4x1 multiplexers each having four data inputs, 0 through 3, and two selection inputs, S1 and S0 
Q.3	(a)	The operation field of an instruction specifies the operation to be performed. This operation will be executed on some data which is stored in computer registers or the main memory. The way any operand is selected during the program execution is dependent on the addressing mode of the instruction. The purpose of using addressing modes is as follows: 1. To give the programming versatility to the user. 2. To reduce the number of bits in addressing field of instruction.

Types of Addressing Modes

Below we have discussed different types of addressing modes one by one:

Immediate Mode

In this mode, the operand is specified in the instruction itself. An immediate mode instruction has an operand field rather than the address field.

For example: ADD 7, which says Add 7 to contents of accumulator. 7 is the operand here.

Register Mode

In this mode the operand is stored in the register and this register is present in CPU. The instruction has the address of the Register where the operand is stored.

Advantages

- Shorter instructions and faster instruction fetch.
- Faster memory access to the operand(s)

Disadvantages

- Very limited address space
- Using multiple registers helps performance but it complicates the instructions.

Register Indirect Mode

In this mode, the instruction specifies the register whose contents give us the address of operand which is in memory. Thus, the register contains the address of operand rather than the operand itself.

Auto Increment/Decrement Mode

In this the register is incremented or decremented after or before its value is used.

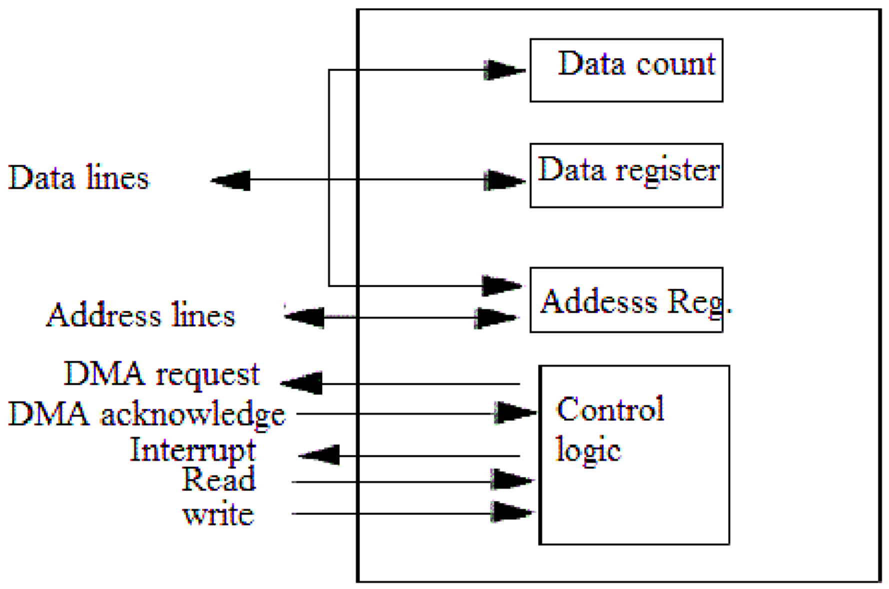
Direct Addressing Mode

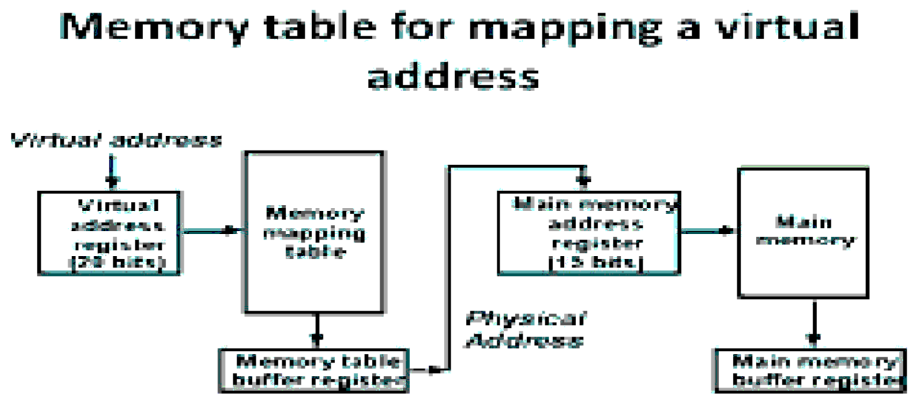
In this mode, effective address of operand is present in instruction itself.

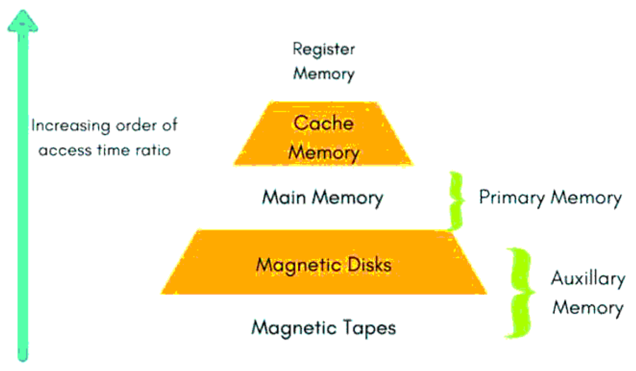
- Single memory reference to access data.
- No additional calculations to find the effective address of the operand.

For Example: ADD R1, 4000 - In this the 4000 is effective address of operand.

		Indirect Addressing Mode In this, the address field of instruction gives the address where the effective address is stored in memory. This slows down the execution, as this includes multiple memory lookups to find the operand. Relative Addressing Mode It is a version of Displacement addressing mode. In this the contents of PC(Program Counter) is added to address part of instruction to obtain the effective address. EA = A + (PC), where EA is effective address and PC is program counter. The operand is A cells away from the current cell(the one pointed to by PC) Base Register Addressing Mode It is again a version of Displacement addressing mode. This can be defined as EA = A + (R), where A is displacement and R holds pointer to base address. Stack Addressing Mode In this mode, operand is at the top of the stack. For example: ADD, this instruction will *POP* top two items from the stack, add them, and will then *PUSH* the result to the top of the stack.
	(b)	**Direct Memory Access** (DMA) transfers the block of data between the *memory* and *peripheral devices* of the system, **without the participation** of the **processor**. The unit that controls the activity of accessing memory directly is called a **DMA controller**. The processor **relinquishes the system bus** for a few clock cycles. So, the DMA controller can accomplish the task of data transfer via the system bus. In this section, we will study in brief about DMA, DMA controller, registers, advantages and disadvantages.

		 Data count Data register Addesss Reg. Control logic Data lines Address lines DMA request DMA acknowledge Interrupt Read write DMA controller is a **hardware unit** that allows I/O devices to access memory directly without the participation of the processor. Here, we will discuss the working of the DMA controller. Below we have the diagram of DMA controller that explains its working Whenever an I/O device wants to transfer the data to or from memory, it sends the DMA request (**DRQ**) to the DMA controller. DMA controller accepts this DRQ and asks the CPU to hold for a few clock cycles by sending it the Hold request (**HLD**). CPU receives the Hold request (HLD) from DMA controller and relinquishes the bus and sends the Hold acknowledgement (**HLDA**) to DMA controller. After receiving the Hold acknowledgement (HLDA), DMA controller acknowledges I/O device **(DACK)** that the data transfer can be performed and DMA controller takes the charge of the system bus and transfers the data to or from memory. When the data transfer is accomplished, the DMA raise an **interrupt** to let know the processor that the task of data transfer is finished and the processor can take control over the bus again and start processing where it has left. Now the DMA controller can be a separate unit that is shared by various I/O devices, or it can also be a part of the I/O device interface.
Q.4	(a)	Virtual memory is a concept used in some large computer systems that permit the user to construct programs as though a large memory space were available, equal to the totality of auxiliary memory. Each address that is referenced by the CPU goes through an address mapping from the so-called virtual address to a physical address in main memory.

		A virtual memory system provides a mechanism for translating program-generated addresses into correct main memory locations. This is done dynamically, while programs are being executed in the CPU. The translation or mapping is handled automatically by the hardware by means of a mapping table. In our example, the address field of an instruction code will consist of 20 bits but physical memory addresses must be specified with only 15 bits. A table is then needed, as shown in Fig. to map a virtual address of 20 bits to a physical address of Auxiliary memory Program 1 Data 1, 1 Data 1, 2 Program 2 Data 2, 1 Program 1 Data 1, 1 Address space N = 1024K = 220 Memory space M = 32k = 215 Main memory 148 15 bits. The mapping is a dynamic operation, which means that every address is translated immediately as a word is referenced by CPU. The mapping table may be stored in a separate memory as shown in Fig. 7-17 or in main memory. In the first case, an additional memory unit is required as well as one extra memory access time. In the second case, the table.  Memory table for mapping a virtual address Virtual address Virtual address register (20 bits) Memory mapping table Memory table buffer register Physical Address Main memory address register (15 bits) Main memory Main memory buffer register
	(b)	Memory Hierarchy

		 Register Memory Increasing order of access time ratio Cache Memory Main Memory Primary Memory Magnetic Disks Auxillary Memory Magnetic Tapes The total memory capacity of a computer can be visualized by hierarchy of components. The memory hierarchy system consists of all storage devices contained in a computer system from the slow Auxiliary Memory to fast Main Memory and to smaller Cache memory. **Auxillary memory** access time is generally **1000 times** that of the main memory, hence it is at the bottom of the hierarchy. The **main memory** occupies the central position because it is equipped to communicate directly with the CPU and with auxiliary memory devices through Input/output processor (I/O). When the program not residing in main memory is needed by the CPU, they are brought in from auxiliary memory. Programs not currently needed in main memory are transferred into auxiliary memory to provide space in main memory for other programs that are currently in use. The **cache memory** is used to store program data which is currently being executed in the CPU. Approximate access time ratio between cache memory and main memory
Q.5	(a)	**Instruction pipelining** is a technique used in the design of modern microprocessors, microcontrollers and CPUs to increase their instruction throughput (the number of instructions that can be executed in a unit of time). The main idea is to divide (termed "split") the processing of a CPU instruction, as defined by the instruction **microcode**, into a series of independent steps of micro-operations (also called **"microinstructions"**, **"micro-op"** or **"µop"**), with storage at the end of each step. This allows the CPUs control logic to handle instructions at the processing rate of the slowest step, which is much faster than the time needed to process the instruction as a single step. The term pipeline refers to the fact that each step is carrying a single microinstruction (like a drop of water), and each step is linked to another step (analogy; similar to water pipes). Most modern CPUs are driven by a clock. The CPU consists internally of logic and

memory (flip flops). When the clock signal arrives, the flip flops store their new value then the logic requires a period of time to decode the flip flops new values. Then the next clock pulse arrives and the flip flops store another values, and so on. By breaking the logic into smaller pieces and inserting flip flops between pieces of logic, the time required by the logic (to decode values till generating valid outputs depending on these values) is reduced. In this way the clock period can be reduced. For example, the RISC pipeline is broken into five stages with a set of flip flops between each stage as follows:

1. Instruction fetch
2. Instruction decode and register fetch
3. Execute
4. Memory access
5. Register write back

Processors with pipelining consist internally of stages (modules) which can semi-independently work on separate microinstructions. Each stage is linked by flip flops to the next stage (like a "chain") so that the stage's output is an input to another stage until the job of processing instructions is done. Such organization of processor internal modules reduces the instruction's overall processing time.

(b) There are some factors that cause the pipeline to deviate its normal performance. Some of these factors are given below:

1. Timing Variations

All stages cannot take same amount of time. This problem generally occurs in instruction processing where different instructions have different operand requirements and thus different processing time.

2. Data Hazards

When several instructions are in partial execution, and if they reference same data then the problem arises. We must ensure that next instruction does not attempt to access data before the current instruction, because this will lead to incorrect results.

3. Branching

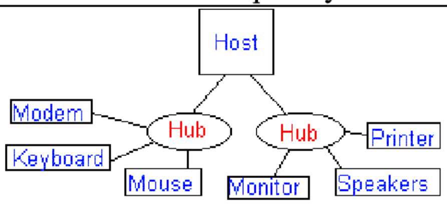
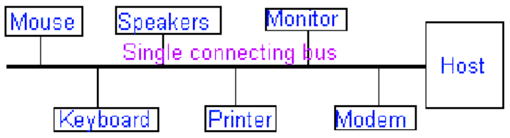
In order to fetch and execute the next instruction, we must know what that instruction is. If the present instruction is a conditional branch, and its result will lead us to the next instruction, then the next instruction may not be known until the current one is processed.

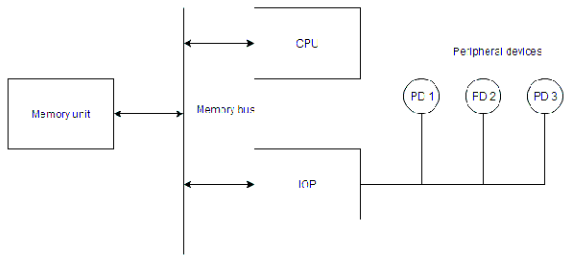
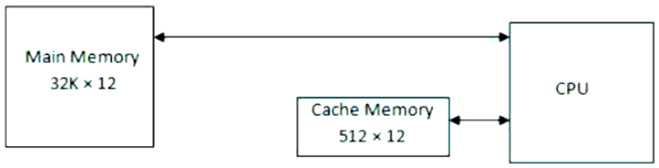
4. Interrupts

Interrupts set unwanted instruction into the instruction stream. Interrupts effect the execution of instruction.

		5. Data Dependency It arises when an instruction depends upon the result of a previous instruction but this result is not yet available.
Q.6	(a)	The memory unit that communicates directly with the CPU is called the main memory. The main memory is the central storage unit in a computer system. It is a relatively large and fast memory used to store programs and data during the computer operation. The principal technology used for the main memory is based on semiconductor integrated circuits. Integrated circuit RAM chips are available in two possible operating modes, static and dynamic. The static RAM consists essentially of internal flip-flops that store the binary information. The stored information remains valid as long as power is applied to unit. The dynamic RAM stores the binary information in the form of electric charges that are applied to capacitors. The capacitors are provided inside the chip by MOS transistors. The stored charge on the capacitors tend to discharge with time and the capacitors must be periodically recharged by refreshing the dynamic memory. Refreshing is done by cycling through the words every few milliseconds to restore the decaying charge. The dynamic RAM offers reduced power consumption and larger storage capacity in a single memory chip. The static RAM is easier to use and has shorted read and write cycles. Both are temporary memories but they vary mainly based on speed, size and cost. 1. Placement: Cache is usually present on the CPU chip itself. Primary memory(RAM) is placed on the motherboard and is connected to the CPU via the Memory Bus. 2. Speed: Because cache is closer to the CPU, it is much faster than RAM. Each read access on the primary memory has to travel via the Memory Bus while the CPU cache is right there. 3. Size: The size of the cache is much less compared to that of primary memory. The size of Primary Memory or RAM in today's computers is a few GBs while the size of cache is a few MBs. 4. Cost: Cache is more expensive than primary memory. Why to have another temporary memory when we already have cheap and large main memory? It is mainly to improve speed.

		The cache is there to reduce the average memory access time for the CPU. When the CPU needs some data from the memory, the cache is checked first and if data is available in the cache it gets it from there. There is no need to perform a memory read. Sometimes cache is divided into two levels called L1 cache(Level 1) and L2 cache(Level 2). L1 is the closest to CPU and also the most expensive. Whenever there is a memory read request, L1 is checked first and then L2 and then the primary memory.
	(b)	There are two main types of software: systems software and application software. Systems software includes the programs that are dedicated to managing the computer itself, such as the operating system, file management utilities, and disk operating system (or DOS). System software is a software that provides platform to other softwares. Some examples can be operating systems, antivirus softwares, disk formating softwares, Computer language translators etc. These are commonly prepared by the computer manufacturers. These softwares consists of programs written in low-level languages, used to interact with the hardware at a very basic level. System software serves as the **interface between the hardware and the end users.** **The most important features of system software include :** 1. Closeness to the system 2. Fast speed 3. Difficult to manipulate 4. Written in low level language 5. Difficult to design
Q.7	(a)	Pipelining a CPU improves its throughput. More pipeline stages means each stage can run faster, increasing the clock speed of the chip. Having more pipeline stages increases the number of instructions that can be executed per time. However, the latency of individual instructions will be increased because of the added pipeline registers. Also, there is an increased area and power cost for the pipeline registers. One big issue with increasing pipeline stages is that it is only effective if you have instructions to feed the pipeline. For example, dependencies can be a challenge for deeply pipelined processors. Unless dependencies can be handled through bypassing, stalls/pipeline bubbles may be introduced which decrease the throughput achieved. Also, branch mispredicts can become very costly as a deeper pipeline can also mean more "wasted" computation on a branch mispredict. The misprediction also lowers the achieved throughput from the maximum possible. The speedup(S) of a pipeline processing over an equivalent non-pipeline processing is defined by the ratio:

		$S = ntn/(k+n-1)tp$ As the number of tasks increases, n becomes much larger than $k-1$, and $k+n-1$ approaches the value of n. Under this condition, the speedup becomes: $S = tn/tp$
	(b)	USB is a system for connecting a wide range of peripherals to a computer, including pointing devices, displays, and data storage and communications products. Although not a relatively new development in personal computing, it has only recently gained popularity due to increasing software support. This document discusses what the USB system does and how it is done. The technical detail covers the system's logical structure, rather than the electrical or software characteristics. The Universal Serial Bus is a network of attachments connected to the host computer. These attachments come in two types known as Functions and Hubs. ***Functions*** are the peripherals such as mice, printers, etc. ***Hubs*** basically act like a double adapter does on a power-point, converting one socket, called a ***port***, into multiple ports. Hubs and functions are collectively called ***devices***. As far as the functions are concerned, hubs are furthermore like double adapters because although the entire system is physically in the star topology seen in Figure 1(a), logically the system acts like a bus topology. This means that signals appear to travel along a single set of wiring, called the ***bus***, to the host and is accessible by all functions, as illustrated in Figure 1(b). However, the host does keep track of the physical arrangements so that if a hub becomes disconnected, it is aware that all hubs and functions attached to it will consequently be disconnected too.  Host Modem Hub Hub Printer Keyboard Mouse Monitor Speakers Figure 1(a) : The physical USB arrangement Functions are joing to hubs in a star arrangement  Mouse Speakers Monitor Single connecting bus Host Keyboard Printer Modem Figure 1(b) : How the USB system appears to functions
Q.8	(i)	An input-output processor (IOP) is a processor with direct memory access capability. In this, the computer system is divided into a memory unit and number of processors. Each IOP controls and manage the input-output tasks. The IOP is similar to CPU except that it handles only the details of I/O processing. The IOP can fetch and execute its own instructions. These IOP instructions are designed to manage I/O transfers only. The IOP operates independent from CPU and transfer data between peripherals and memory.

		 CPU Peripheral devices PD 1 PD 2 PD 3 Memory unit Memory bus IOP
	(ii)	If the active portions of the program and data are placed in a fast small memory, the average memory access time can be reduced, thus reducing the total execution time of the program. Such a fast small memory is referred to as a cache memory. It is placed between the CPU and main memory as illustrated in Figure. The cache memory access time is less than the access time of main memory by a factor of 5 to 10. The cache is the fastest component in the memory hierarchy and approaches the speed of CPU components. A system with 512 x 12 cache and 32 K x 12 of main memory. 
8	(iii)	**Bus** A typical digital computer has many registers, and paths must be provided to transfer in formation form one register to another. The number of wires will be excessive if separate lines are used between each register and all other registers in the system. A more efficient scheme for transferring information between registers in a multiple-register configuration is a common bus system. A bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time. Control signals determine which register is selected by the onus during each particular register transfer. **Memory Transfer** The transfer of information from a memory word to the outside environment is called a read operation. The transfer of new information to be stored into the memory is called a write operation. A memory word will be symbolized by the letter M. The particular memory word among the many available is selected by the memory address during he transfer. It is necessary to specify the address of M when writing memory transfer operations; this will be done by enclosing the address in square brackets following the letter M. Consider a memory unit that receives the address form a register, called the address register, symbolized by AR. The data are transferred to another register, called the data register, symbolized by DR. the read operation can be stated as follows: Read: DR ← M [AR]

8 (iv)

Basis For Comparison	**UMA**	**NUMA**
Basic	Uses a single memory controller	Multiple memory controller
Type of buses used	Single, multiple and crossbar.	Tree and hierarchical
Memory accessing time	Equal	Changes according to the distance of microprocessor.
Suitable for	General purpose and time-sharing applications	Real-time and time-critical applications
Speed	Slower	Faster
Bandwidth	Limited	More than UMA.