1.

An Interrupt occurs when an external source request the CPU access. If an interrupt occurs while the instruction is being executed it is held until the execution of the instruction is completed and before fetching the next instruction. The CPU will handle the external interrupt first and resumes back the operations after that. If more than two interrupts occur at the same time we can 1) **Disable Interrupt**, we can disable all the other interrupts while the current interrupt is being processes. 2) We can assign priority to the interrupts and we can follow either the follow the preemptive approach or non-preemptive approach while handling a higher priority over the lower priority. An interrupt of higher priority could interrupt a lower priority interrupt. If an interrupt occur during an interrupt service routine we can follow the **nested interrupt** service subroutine approach. If an interrupt of lower priority tries to interrupt a higher priority then we will keep the lower priority interrupt aside. If an interrupt of higher priority interrupts a lower priority then the lower priority ISR is interrupted and the state is pushed on to the stack, and execution continues with higher priority ISR.

2a.

Advantages are as follows

1. All the components are connected to the same bus hence it has a shared transmission medium.
2. We can add new components easily to the bus
3. It is simple in operation and the cost is low as well.

Disadvantages of using a single bus are

1. If two devices transmit during the same time period, their signals will overlap and become garbled. Thus, only one device at a time can successfully transmit.
2. if there is any fault occur, all system affected and the other feeders pass over.
3. High Latency and low data rates.

2b.

The **peripheral component interconnect** is a popular high-bandwidth, processor-independent bus that can function as a mezzanine or peripheral bus. Compared with other common bus specifications, PCI delivers better system performance for high-speed I/O subsystems. A mezzanine architecture requires only a bridge between the processor's bus and the high-speed bus. There is a local bus that connects the processor to a cache controller, which is in turn connected to a system bus that supports main memory. This bus supports connections to high-speed LAN's, as well as interface controllers to local peripheral buses. The high-speed bus arrangement specifically is designed to support high capacity I/O devices. Lower-speed devices are still supported off an expansion bus, with an interface buffering traffic between the expansion bus and the high-speed bus.

2c.

Bus arbitration is a process by which next device becomes the bus controller by transferring bus master-ship to another bus. A device that initiates data transfers on the bus at any given time is called a bus master. Because all the components are exchanging data on a shared medium, it need arbitration to assure only one component transmit data at a time. There are two types of Bus Arbitration.

Centralized arbitration:

Only a single bus arbiter performs the required arbitration, it can be either a processor or a separate DMS controller.

There are three different arbitration schemes that use the centralized bus
arbitration approach.
1. Daisy chaining : All the masters use the same line for making bus requests.
2. Polling method: The controller is used to generate address lines for the master
3. Independent request: Each bus has its own bus request and a grant
• Distributed arbitration:
No single arbiter, all devices participate in the selection of the next bus
master. Each module may claim the bus, each device on the bus is assigned a 4-
bit identification number. When one or more devices request control of the bus, they assert
the start arbitration signal and place their 4-bit identification numbers on
arbitration lines through ARB3. Each device compares the code and changes its bit position
accordingly.

3.1.

Given the instruction size is 48 bits, where opcode size is 16 bits. Therefore the address bits will be 48 - 16 = 32 bits. The maximum directly addressable memory capacity will be $2^{32}$ bytes. Which is **4GB**

3.2.

Address size is 48 - 16, which is 32bits. Program Counter needs 32 bits.
Instruction size is 48 bits. The Instruction Register should be 48bits in size.

3.3

The impact on the system speed will be as follows.

i) If address bus is 32 bits and data bus is 16 bits.

We know that maximum addressable memory capacity is 32 bits. So 1 cycle will be enough to read or write on the address bus.
Since memory is 32-bit in size, we need 2 clock cycles to read/write from memory on data bus of 16 bits.
Since instruction is 48bits, we will require 3 clock cycles to read/write instruction using the data bus.

ii) If address bus is 32 bits and data bus is 32 bits.

We know that maximum addressable memory capacity is 32 bits. So 1 cycle will be enough to read or write on the address bus.
Since memory is 32-bit in size, we need 1 clock cycles to read/write from memory on data bus of 32 bits.
Since instruction is 48bits, we will require 2 clock cycles to read/write instruction using the data bus of 32 bits.

We can say that speed of case(ii) is greater than case(i) since we require more cycles in case i to read/write data and instruction.

4. I worked on question 4, since we need not turn in question 4, I am not including it in my assignment.

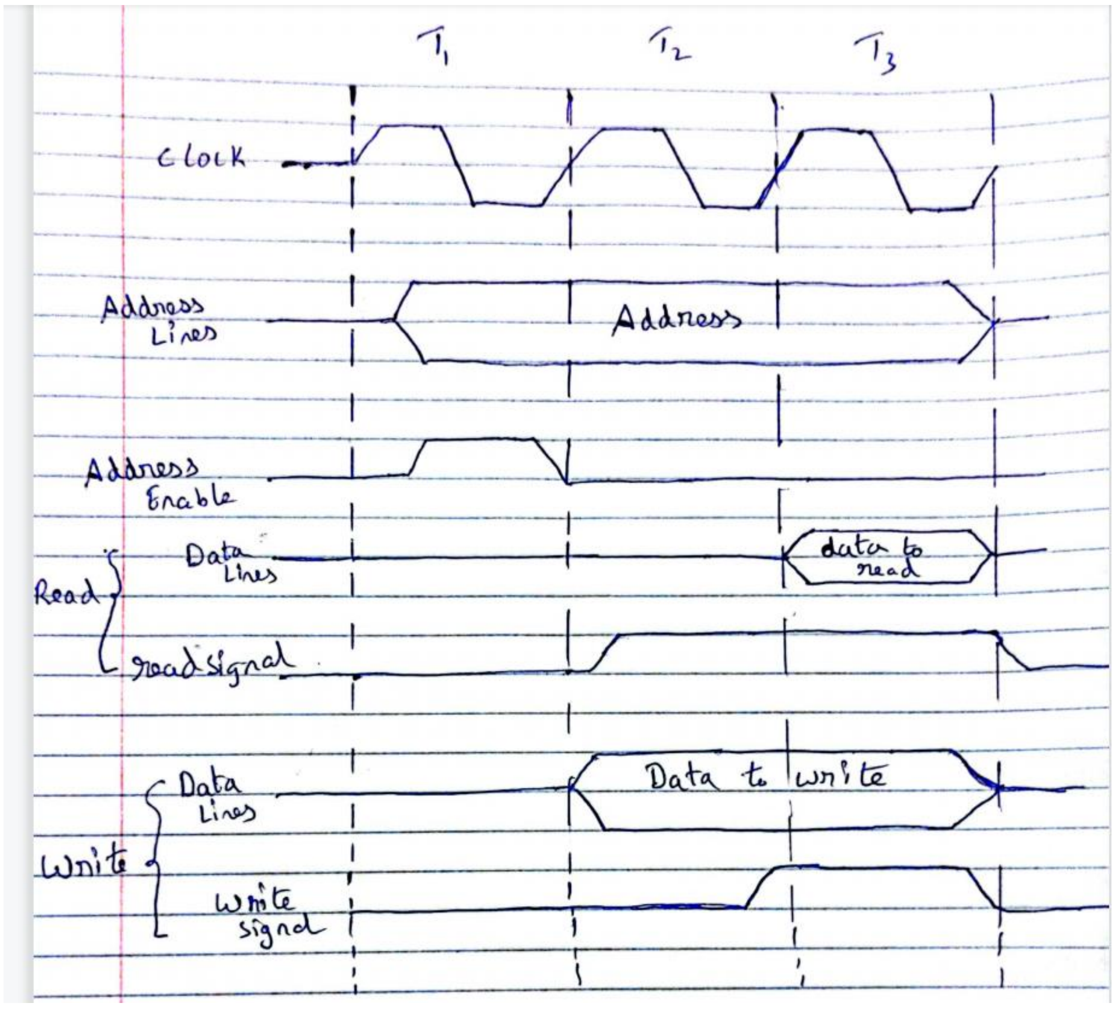
5.

Given 16 bit external data bus and clock frequency of 5GHZ. The bus cycle equal to 5 input clock cycles which is (5 / 5GHZ)1ns.

Maximum data transfer rate = 16 bits / 1 ns = 16 Gbps.

If we increase the external data bus to 32bits or double the clock frequency, in both the cases the maximum data transfer rate will be doubled to 32Gbps. The bottleneck is that if we increase the external data bus to 32 bits, this will require memory size to be 32 bits as well to achieve performance. If we double the clock frequency the memory access rates should be improved as well otherwise the processor will have to wait for memory access. It is easy to increase the external data bus size without much complications, but if we increase the clock frequency we need every activity to match the clock speed which might be difficult.

6.



With synchronous timing, the occurrence of events on the bus is determined by a clock. The sending and receiving units are enabled with the same clock signal. Let us consider the above example. We have three clock cycles, in the first clock cycle the address enable signal is high and we have the address from where we need to read or write on the address lines. For the read operation the read signal is enabled in the second clock cycle and after one clock cycle we will read the data from the address onto the data lines as we can see above. For the write operation we will have the data ready to write in a particular memory location at the start of the second clock cycle, once the write signal is enabled in the third clock cycle, the data is written from data lines to a particular memory location.

7.

Initially if the highest priority agent passes the BPR signal and before it reaches the lowest priority agent, if the clock cycle got completed than there will be more than one bus master trying to take control of the bus. The master with low priority has not come to know that the master with high priority is trying to take the bus control. As a result there will be attempts by multiple masters to take control over the bus and this will lead to bad outcomes as bus can only be accessed by one master at a time. To overcome this the amount of the desired time taken for BPR signal to propagate  from greater preference agent to lower preference agent must be Less than the desired clock cycle.