QUESTION -2

a. What do you mean by memory-mapped I/O and I/O mapped I/O? Give example.

Answer:

memory-mapped I/O:

In memory mapping of I/O devices, the I/O ports are assigned 16-bit addresses within the memory. Here each bus is common thus the same set of instructions is used for memory and I/O devices. Thus, I/O is considered as memory and the same address space is used by both memory and I/O devices. This reduces the addressing capability of the memory.

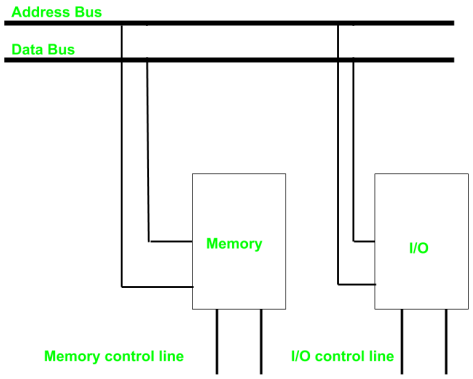
In this case, the processor considers the I/O ports as memory locations for the purpose of reading and writing. So, whenever, an address is generated on the address bus then a resultantly control signal is generated for memory read. In such a case, the processor is not concerned whether the responding data is coming from a memory device or an I/O device. The same is the case with the memory write operation

I/O mapped I/O:

It is also known as **Isolated I/O mapping** and the reason for the same is that here the address space of memory and I/O are separated from each other. Thus, different read and write instructions are used for I/O and memory. In this approach, there is a common bus for I/O devices and memory however, individual read and write control lines are used for I/O.

Here the operation takes place in a way that, if the data over which operation is to be performed is to be collected from the I/O devices then address is placed on the address line and I/O read and I/O write control lines will get activated so that data transfer can be performed between the processor and I/O.

For the transfer of data between the processor and I/O devices, only IN and OUT instructions are used in the isolated mapping. The required chip select signals in this case are generated by an individual decoding unit.



C. Define instruction pipelining.

ANSWER:

the processor processes each instruction from the start till it finishes execution and then it fetches the next instruction for instruction. Now if we divide the instruction execution into stages then it could be divided into:

- **Instruction fetches:** Fetch the instruction from the main memory or cache.
- **Instruction decoding:** The processor interprets instruction and determines the operation that has to be performed.
- **Operand fetch:** If the execution of the instruction requires operand then the processor fetches operand from the main memory or cache.
- **Instruction Execution:** The processor performs the desired operation.
- **Operand Store:** The result of execution is stored.

Now each stage is handled by the different sections of the hardware. So, once the instruction is fetched by the fetching unit of the hardware and is forwarded to other hardware sections for decoding, operand fetches, execution, and

operand store; the fetching unit of hardware has to sit idle till the operand store stage is completed