Memory mapped I/O and Isolated I/O

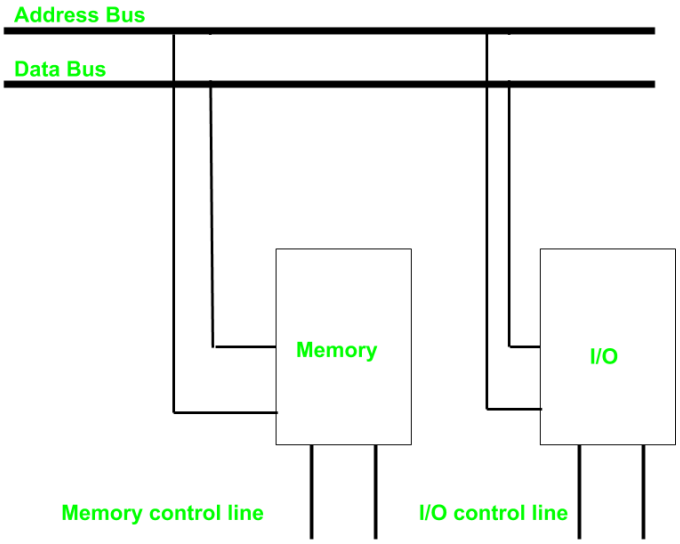
As a CPU needs to communicate with the various memory and input-output devices (I/O) as we know data between the processor and these devices flow with the help of the system bus. There are three ways in which system bus can be allotted to them :

1. Separate set of address, control and data bus to I/O and memory.
2. Have common bus (data and address) for I/O and memory but separate control lines.
3. Have common bus (data, address, and control) for I/O and memory.

In first case it is simple because both have different set of address space and instruction but require more buses.

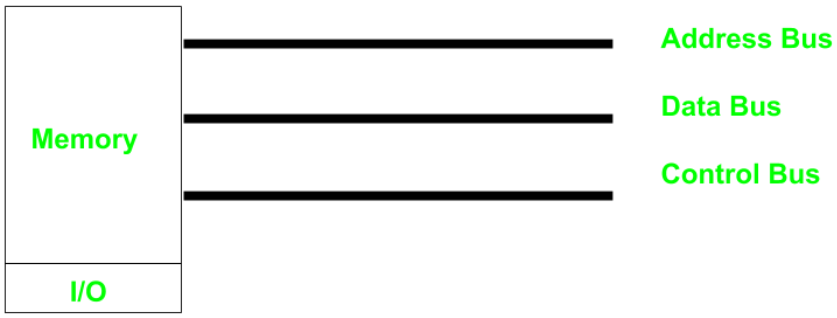
Isolated I/O –

Then we have Isolated I/O in which we Have common bus(data and address) for I/O and memory but separate read and write control lines for I/O. So when CPU decode instruction then if data is for I/O then it places the



address on the address line and set I/O read or write control line on due to which data transfer occurs between CPU and I/O. As the address space of memory and I/O is isolated and the name is so. The address for I/O here is called ports. Here we have different read-write instruction for both I/O and memory.

Memory Mapped I/O –



In this case every bus in common due to which the same set of instructions work for memory and I/O. Hence we manipulate I/O same as memory and both have same address space, due to which addressing capability of memory become less because some part is occupied by the I/O.**Differences between memory mapped I/O and isolated I/O –**

Isolated I/O	Memory Mapped I/O
Memory and I/O have separate address space	Both have same address space
All address can be used by the memory	Due to addition of I/O addressable memory become less for memory
Separate instruction control read and write operation in I/O and Memory	Same instructions can control both I/O and Memory
In this I/O address are called ports.	Normal memory address are for both
More efficient due to separate buses	Lesser efficient
Larger in size due to more buses	Smaller in size
It is complex due to separate logic is used to control both.	Simpler logic is used as I/O is also treated as memory only.

Advantages of Memory-Mapped I/O:

Faster I/O Operations: Memory-mapped I/O allows the CPU to access I/O devices at the same speed as it accesses memory. This means that I/O operations can be performed much faster compared to isolated I/O.

Simplified Programming: Memory-mapped I/O simplifies programming as the same instructions can be used to access memory and I/O devices. This means that software developers do not have to use specialized I/O instructions, which can reduce programming complexity.
Efficient Use of Memory Space: Memory-mapped I/O is more memory-efficient as I/O devices share the same address space as the memory. This means that the same memory address space can be used to access both memory and I/O devices.

Disadvantages of Memory-Mapped I/O:

Limited I/O Address Space: Memory-mapped I/O limits the I/O address space as I/O devices share the same address space as the memory. This means that there may not be enough address space available to address all I/O devices.
Slower Response Time: If an I/O device is slow to respond, it can delay the CPU's access to memory. This can lead to slower overall system performance.

Advantages of Isolated I/O:

Large I/O Address Space: Isolated I/O allows for a larger I/O address space compared to memory-mapped I/O as I/O devices have their own separate address space.
Greater Flexibility: Isolated I/O provides greater flexibility as I/O devices can be added or removed from the system without affecting the memory address space.
Improved Reliability: Isolated I/O provides better reliability as I/O devices do not share the same address space as the memory. This means that if an I/O device fails, it does not affect the memory or other I/O devices.

Disadvantages of Isolated I/O:

Slower I/O Operations: Isolated I/O can result in slower I/O operations compared to memory-mapped I/O as it requires the use of specialized I/O instructions.

More Complex Programming: Isolated I/O requires specialized I/O instructions, which can lead to more complex programming.

Graphics processing: Memory-mapped I/O is often used in graphics cards to provide fast access to frame buffers and control registers. The graphics data is mapped directly to memory, allowing the CPU to read from and write to the graphics card as if it were accessing regular memory.
Network communication: Network interface cards (NICs) often utilize memory-mapped I/O to transfer data between the network and the system memory. The NIC registers are mapped to specific memory addresses, enabling efficient data transfer and control over network operations.
Direct memory access (DMA): DMA controllers employ memory-mapped I/O to facilitate high-speed data transfers between devices and system memory without CPU intervention. By mapping the DMA controller registers to memory, data can be transferred directly between devices and memory, reducing CPU overhead.

Isolated I/O applications:

Embedded systems: Isolated I/O is commonly used in embedded systems where strict isolation between the CPU and peripherals is necessary. This includes applications such as industrial control systems, robotics, and automotive electronics. Isolation ensures that any faults or malfunctions in peripheral devices do not affect the stability of the entire system.
Microcontrollers: Microcontrollers often rely on isolated I/O to interface with various peripherals, such as sensors, actuators, and displays. Each peripheral is assigned a separate I/O port, allowing the microcontroller to control and communicate with multiple devices independently.
Real-time systems: Isolated I/O is preferred in real-time systems that require precise timing and deterministic behavior. By isolating the I/O operations, these systems can maintain strict control over the timing and synchronization of external events, ensuring reliable and predictable performance.