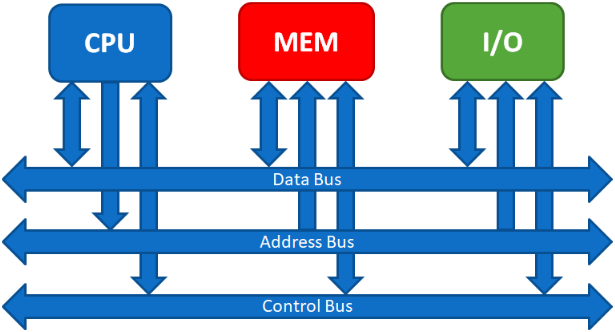
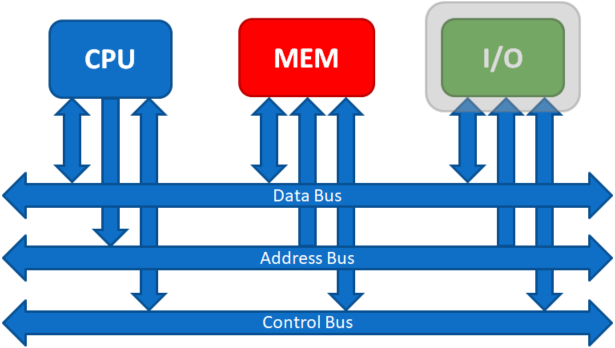
Module 6

Programmed and interrupt-driven IO

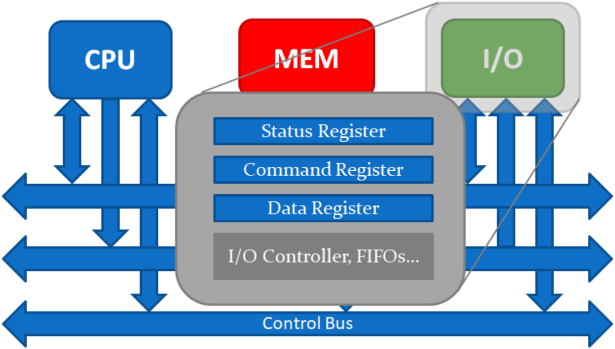
System Bus



System Bus



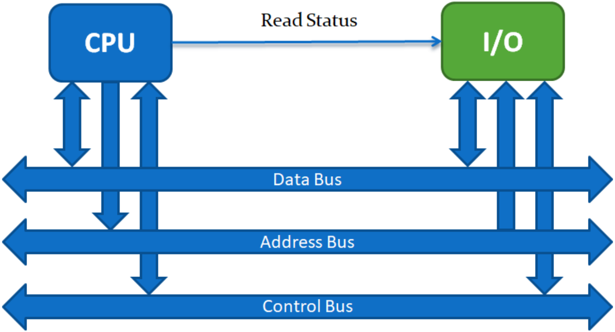
System Bus


CPU
MEM
I/O
Status Register
Command Register
Data Register
I/O Controller, FIFOs...
Control Bus

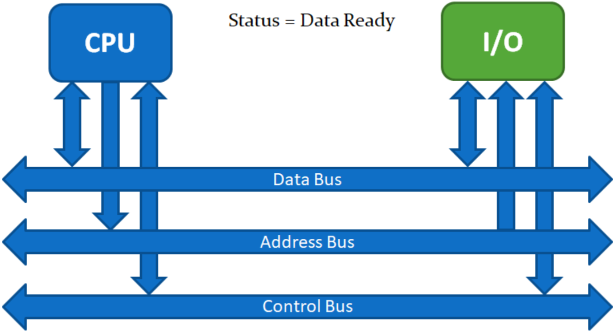
Programmed IO (1)

- CPU periodically tests the status bit of an IO device (aka Polling) to check for readiness of the device
- When device is ready, CPU reads/writes data
- Advantages:
  - Simple to implement
  - Generally offers better determinism
- Disadvantages:
  - CPU might be locked in a waiting state for a long period
  - The response time might be on unacceptable level

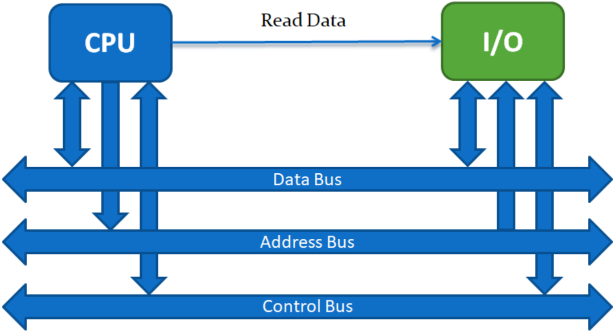
Programmed IO (2)



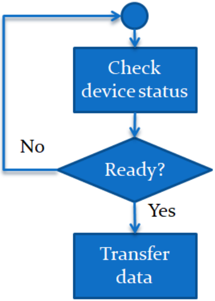
Programmed IO (2)

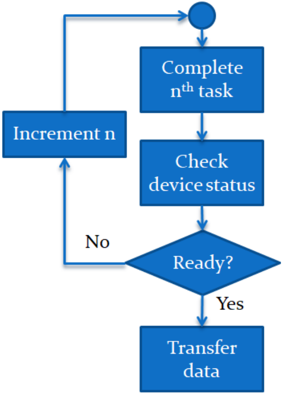


Programmed IO (2)



Programmed IO (3)





Programmed IO (4)

- Pseudo code

Read

```
while (!Data_Ready());
value = Get_Data();
```

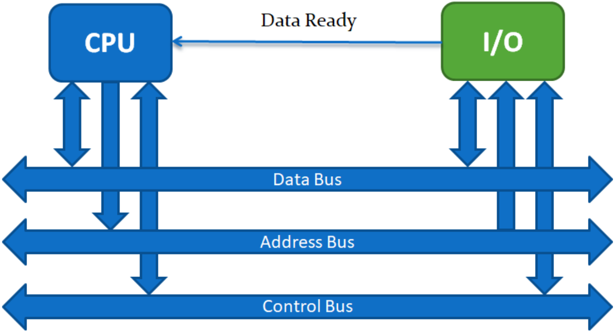
Write

```
while (!Device_Busy());
Put_Data(value);
```

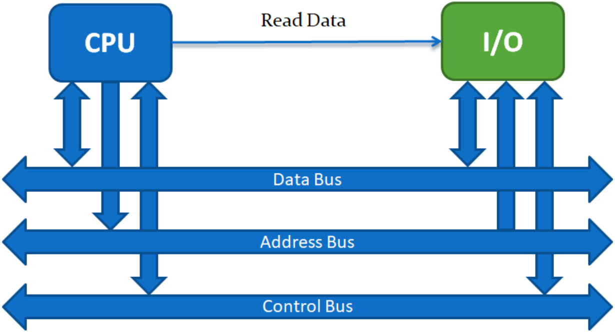
Interrupt-driven IO (1)

- IO device requests attention from the CPU via the interrupt mechanism
- The current state is preserved before servicing an interrupt (context switching) – introduces latency
- Advantages:
  - CPU can do other useful work when an IO device does not require attention
  - Fast response time (in most cases)
- Disadvantages:
  - Low determinism
  - Overhead issues with frequent requests

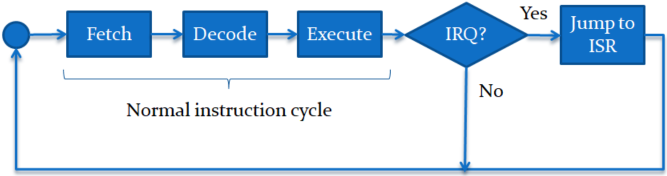
Interrupt-driven IO (2)



Interrupt-driven IO (2)



Interrupt-driven IO (3)



Interrupt-driven IO (4)

- Pseudo code

Main program

```
do_some_work();
```

Interrupt Service Routine (ISR)

```
void Device_ISR() at ISR_addr
{
          value = Get_Data();
}
```