5. Write short notes on inter process communication mechanisms. (16) [CO3-L1]

- ☐ Processes often need to communicate with each other. **Interprocess communication mechanisms** are provided by the operating system as part of the process abstraction.

- ☐ In general, a process can send a communication in one of two ways: **blocking** or **onblocking**.

  After sending a blocking communication, the process goes into the waiting state until it receives a response.

- ☐ Nonblocking communication allows the process to continue execution after sending the communication. Both types of communication are useful. There are two major styles of interprocess communication: **shared memory** and **message passing**.

Shared Memory Communication:

- ☐ Figure 3.9 illustrates how shared memory communication works in a bus-based system. Two components, such as a CPU and an I/O device, communicate through a shared memory location. The software on the CPU has been designed to know the address of the shared location.

- ☐ The shared location has also been loaded into the proper register of the I/O device. If, as in the figure, the CPU wants to send data to the device, it writes to the shared location. The I/O device then reads the data from that location. The read and write operations are standard and can be encapsulated in a procedural interface.

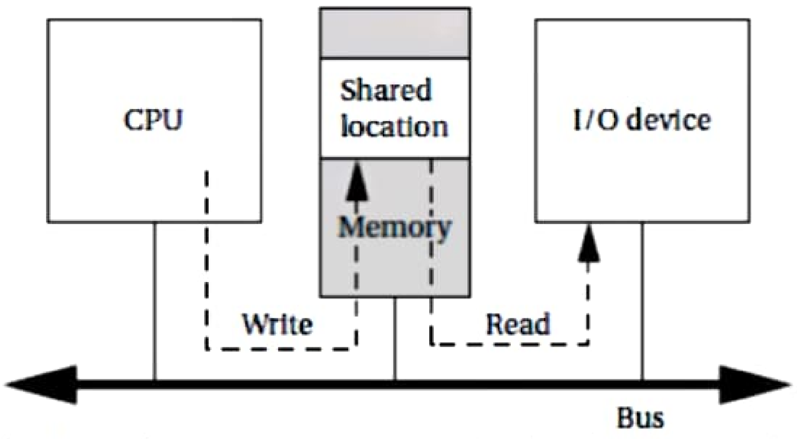

Fig 3.9 Shared memory communication implemented on a bus.

☐As an application of shared memory, let us consider the situation of Figure 6.14in which the CPU and the I/O device want to communicate through a shared memory block. There must be a flag that tells the CPU when the data from the I/O device is ready.

☐The flag, an additional shared data location, has a value of 0 when the data are not ready and 1 when the data are ready. If the flag is used only by the CPU, then the flag can be implemented using a standard memory write operation. If the same flag is used for bidirectional signaling between the CPU and the I/O device, care must be taken. Consider the following scenario:

1. CPU reads the flag location and sees that it is 0.
2. I/O device reads the flag location and sees that it is 0.
3. CPU sets the flag location to 1 and writes data to the shared location.
4. I/O device erroneously sets the flag to 1 and overwrites the data left by the CPU.

Message Passing:

- Message passing communication complements the shared memory model. As shown in Figure, each communicating entity has its own message send/receive unit. The message is not stored on the communications link, but rather at the senders/ receivers at the end points.

  - In contrast, shared memory communication can be seen as a memory block used as a communication device, in which all the data are stored in the communication link/memory.

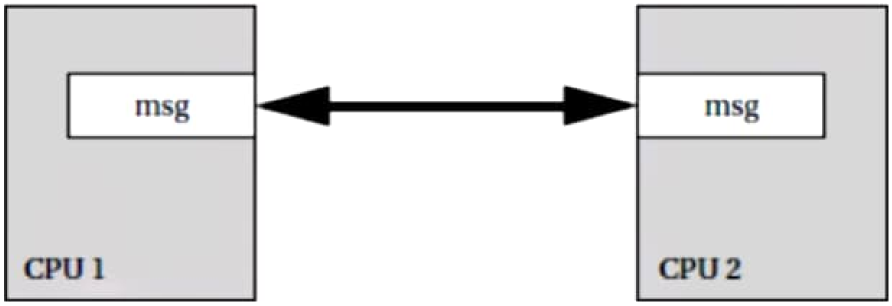

Fig Message passing communication.

- Applications in which units operate relatively autonomously are natural candidates for message passing communication. For example, a home control system has one microcontroller per household device—lamp, thermostat, faucet, appliance, and so on.
- The devices must communicate relatively infrequently; furthermore, their physical separation is large enough that we would not naturally think of them as sharing a central pool of memory.
- Passing communication packets among the devices is a natural way to describe coordination between these devices. Message passing is the natural implementation of communication in many 8-bit microcontrollers that do not normally operate with external memory.

Signals

Another form of interprocess communication commonly used in Unix is the **signal**. A signal is simple because it does not pass data beyond the existence of the signal itself. A signal is analogous to an interrupt, but it is entirely a software creation. A signal is generated by a process and transmitted to another process by the operating system.

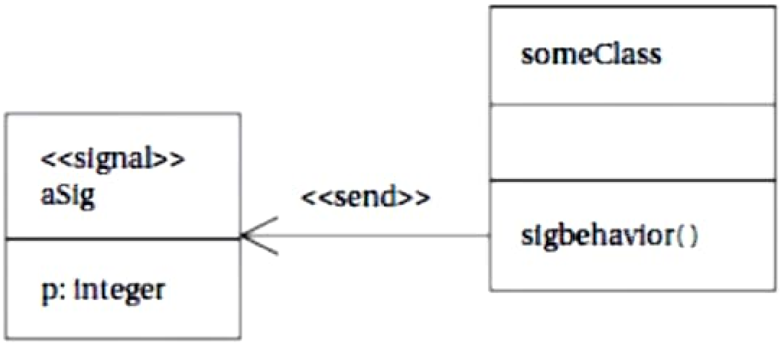

Fig Use of a UML signal.

A UML signal is actually a generalization of the Unix signal. While a Unix signal carries no parameters other than a condition code, a UML signal is an object. As such, it can carry parameters as object attributes. Figure 3.11 shows the use of a signal in UML. The sigbehavior ( ) behavior of the class is responsible for throwing the signal, as indicated by <<send>>.The signal object is indicated by the <<signal>> stereotype.