Assignment 2

Computer Organization and Architecture

Aditya Choudhary | 20CS10005

Questions

1. What is the architectural view of an i/o device for a programmer ?
2. How is the architectural view supported organisationally? Explain with the help of a diagram.
3. Consider any simple i/o task (such as reading from a keyboard, reading or writing a block of data to the disk) and explain how that would be achieved in your framework.

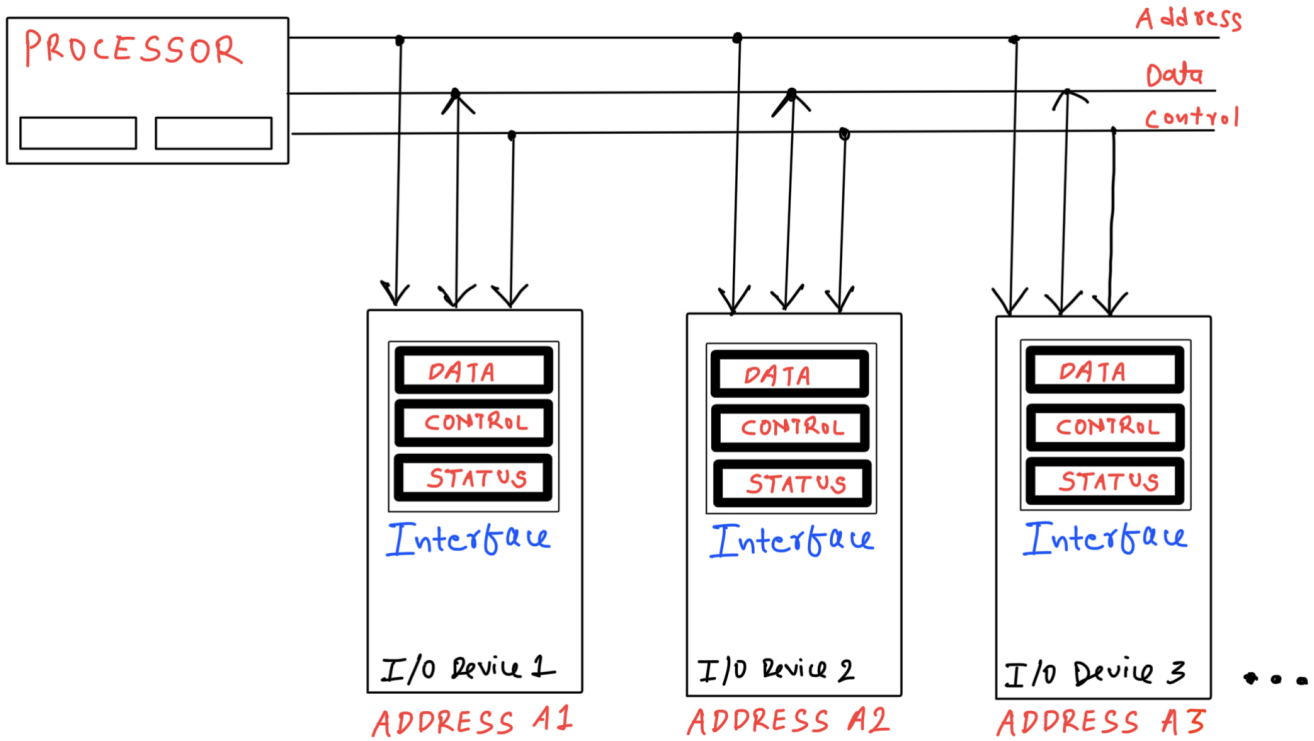
Answer:

1. **What is the architectural view of an i/o device for a programmer ?**

   From the perspective of a programmer, the data from the I/O devices are read using software interrupts. This is made possible because of the fact that the registers in the interfacing modules of various I/O devices have an address associated with them. This makes it possible for the processor to access and modify those registers using instructions in a program. This is known as Program controlled I/O.

   Whenever such an instruction is encountered, the processor polls the status bits of the interfacing module of the corresponding I/O device, whenever this bit is set to 1 by the device, the processor either reads or writes the data to the data register to the interfacing module.

2. **How is the architectural view supported organisationally ? Explain with the help of a diagram.**



In this diagram, each of the I/O device interface has 3 registers, Data, Status and Control register. Each device has its own address Ai, and the address of the three register is Ai, Ai+4 and Ai+8 respectively. If we know the address of a port, we can access the three of its registers to read or write data.

The processor sends the address of the register via the address line, takes or sends the data using the data line, generates the control signal and sends them to the registers via the control line. The address determines what device we are looking for and must me mentioned in the instruction in the program.

Suppose processor encounters the instruction READ Ai. This means that we have to read the data from the data register in the interface of the device at address Ai. But we can only do so when the device sends the data.

So whenever the device inputs the data and places it in the data register, the DATAIN flag of the status register is set to 1. The CPU continuously polls this flag until it encounters 1, and then reads the data from the data register and stores it in the memory. It then goes to the next instruction in the sequence.

This acts as a synchronizing mechanism between the vast difference of speeds of data input from device and CPU execution speed. Similar process takes place for outputting data from CPU.

3. **Consider any simple i/o task (such as reading from a keyboard, reading or writing a block of data to the disk) and explain how that would be achieved in your framework.**

   Let us consider an example of a program that does reading from a keyboard, storing in CPU memory and then storing the data in the disk. To perform I/O transfers, the processor must execute machine instructions that check the state of the status flags and transfer data between the processor and the I/O devices.

   Let us first consider inputting from a keyboard. The the address of data register for keyboard be stored as KDATA and status register bit be KIN. We poll KIN until it becomes 1, then read the data from KDATA and store in the CPU registers. KSTATUS is address of status register for keyboard.
   Pseudocode :

```
READ:
LoadWord        R5, KSTATUS     # loading status reg data in R5
andi            R5, R5, KIN     # if KIN is 0, R5 will also be zero
BranchifZero    R5, READ        # keep on polling KIN
LoadWord        R4, KDATA       # if KIN is set, load KDATA into CPU
```

   Now we will store the data from CPU register to the disk. Similar process will be followed. We poll the MIN bit of the MSTATUS register of the disk memory. Whenever the disk is ready to take data, MIN will be set and data will be stored from CPU memory to MDATA register of memory interface.
   Pseudocode :

```
WRITE:
LoadWord        R5, MSTATUS     # loading status reg data in R5
andi            R5, R5, MIN     # if MIN is 0, R5 will also be zero
BranchifZero    R5, WRITE       # keep on polling MIN
StoreWord       R4, MDATA       # if MIN is set, store R4 into MDATA
```