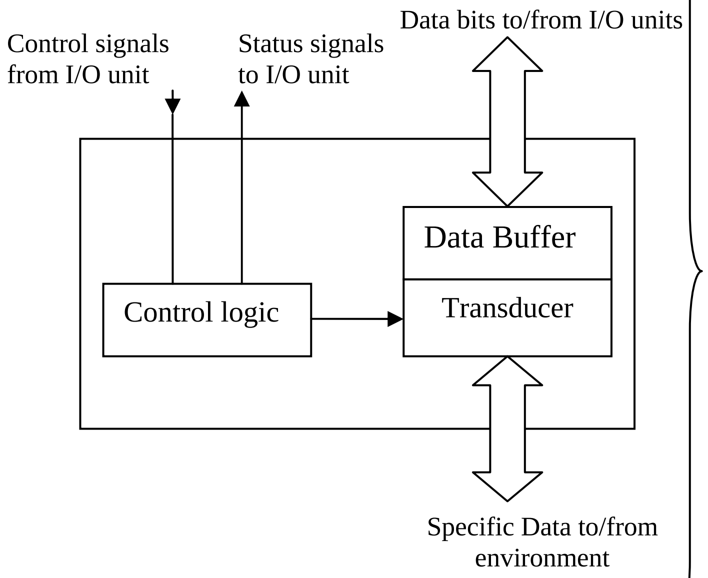
Chapter 6

Organization of the CPU-peripheral devices communication

§1. Peripheral devices in the computer system

- In addition to the CPU and Main Memory, any computer system contains a set of **input/output units** (I/O units). (1.1)
- A powerful computer may have hundreds of **peripheral devices** connected to it through input/output units. (1.2)
- *Each I/O unit (module) interfaces to the system bus and controls one or several peripheral devices.* (1.3)
- *Examples* of **peripheral devices**: keyboard, mouse, video monitor, printer, plotter, scanner, hard-disk driver, CD driver, magnetic disk driver, floppy-disk driver, joy-stick, DVD driver, tape driver. (1.4)
- Peripheral devices exhibit *very large differences* in two important aspects: (1.5)
    - ➢ in *form* and *function*, by using different media, different principles of operation and necessitating different sets of control information.
    - ➢ in *speed* at which they operate, even though the fastest ones are much slower in manipulating information than the CPU.

- Peripheral devices represent one of the **main system resources** that may be shared among multiple users: a line printer may be simultaneously needed by several processes, although they must use it sequentially and a rule of serving has to be established. (1.6)
- The peripheral devices together with I/O units are forming the **I/O architecture** of the computer system.
- The I/O architecture represents the interface to the outside world of a digital computer. (1.7)
- The external devices can broadly be classified in the following groups: (1.8)
    1. *human readable*, that is suitable for communicating with the user.
    2. *machine readable*, that is suitable for communicating with equipment.
    3. *communicating* with remote devices, interchanging data with other computers.
- In a simplified analysis any peripheral device has the following structure: (1.9)


Control signals from I/O unit
Status signals to I/O unit
Data bits to/from I/O units
Data Buffer
Transducer
Control logic
Specific Data to/from environment

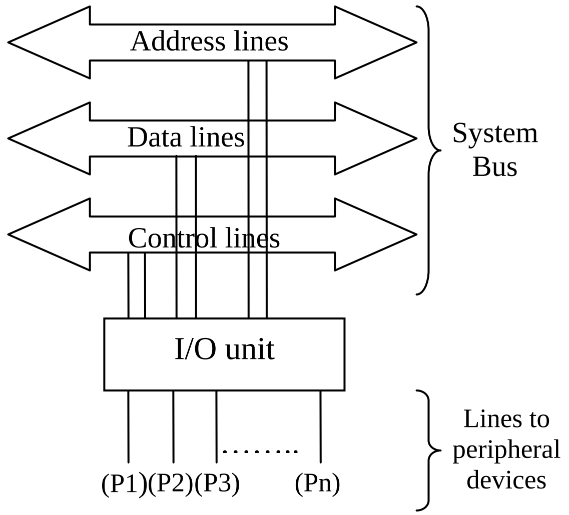
where:

- **Control** signals determine the function that the device will perform.
- **Status** signals indicate the state of the device.
- **Data** are in the form of a set of bits to be sent to or received from the I/O unit.
- **Control Logic** associated with the device controls the device operation in accordance with the commands issued by the I/O unit and the type of peripheral device.
- **Transducer** converts data from the electrical to other forms of energy during output operation and from other forms to electrical during the input operation.
- Typically a **buffer** has to be associated with the Transducer to temporarily hold data being transferred between I/O units and the external environment.

(1.10)

§2. Input/Output units.

- The generic model for an I/O unit is presented below:



(2.1)

- An I/O unit (module) is not simply a set of mechanical connectors that wire a peripheral device into the system bus; it contains some "intelligence" materialized in some logic for performing a *communication function* between the peripheral device and the system bus. (2.2)
- The *necessity of I/O units* is given by the following reasons:
    1. there is a wide variety of peripherals with various methods of operation; therefore, it is impractical to incorporate the necessary logic within the CPU to control all devices.
    2. the data transfer rate of peripherals is much slower than that of the CPU and memory. Thus, it is totally impractical to use the high speed system bus to communicate directly with peripheral devices.
    3. peripherals often use different data formats and word lengths than the CPU to which they are attached.

(2.3)

- Any I/O unit has **two major functions**:
    1. interface to the CPU and memory via the system bus.
    2. interface to one or more peripheral devices.

(2.4)

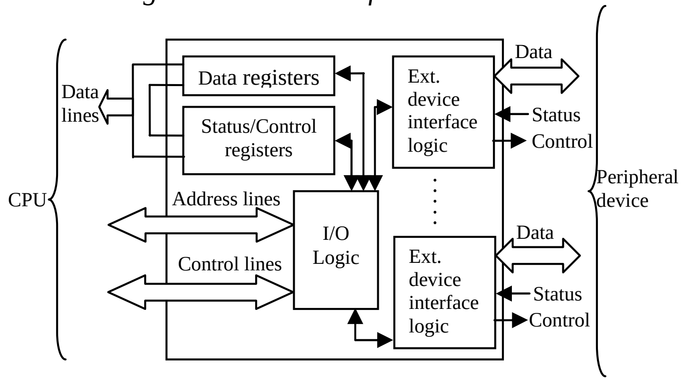
- The major **I/O requirements** are:
    1. Control unit and timing.
    2. CPU communication.
    3. Peripheral device communication
    4. Data buffering
    5. Error detection.

(2.5)

- The CPU may communicate with one or more peripheral devices in unpredictable schemes, depending on the program's need for Input/Output. (2.6)

- In standard organization the system bus must be shared among many activities, including I/O activity. (2.7)

- Description of a **transfer of data** item from a peripheral device to CPU: (2.8)
  1. CPU interrogates the I/O unit to check the status of the device.
  2. The I/O unit returns the device status to CPU.
  3. If the device is operational and ready to transmit, CPU requests the transfer of data item by means of a command sent to the I/O unit.
  4. The I/O unit obtains the data item (8,16,24 bits) from the device.
  5. The data item is transferred from the I/O unit to the CPU.
- Each of the interactions between the CPU and I/O unit involves one or more **bus arbitration**. (2.9)
- I/O units vary considerably in complexity and the number of peripheral devices they control. (2.10)
- The *general structure of an I/O unit* would be as follows:



(2.11)

- Each I/O unit must have a **unique address**.
- If an I/O unit allows connection of several devices, then each device has its own address, so that the I/O unit must be able to recognize and generate addresses of all devices it controls. (2.11)

- Data transfers to and from the I/O units are buffered in one or several **data registers**. (2.12)

- There must be one or several **status registers** that provide *current status information*. (2.13)

- A status register may function also as a **control register** to accept detailed *control information from the CPU*. (2.14)

- CPU uses **control lines** to transmit commands to the I/O units. (2.15)

- In its simplest form, the I/O unit leaves much of the work of controlling a device to the CPU. (2.16)

- I/O unit and the CPU communication assumes the following actions: (2.17)
  1. **Command decoding** – I/O unit accepts commands from the CPU, signals that are sent on Control bus.
  2. **Exchange of data** – carried out over the Data bus.
  3. **Status reporting** – because peripherals are slow it is important to know their status through the I/O unit. Common status signals are BUSY, READY, etc
  4. **Address recognition** – each device must have a unique address so that I/O unit must recognize the unique address for each peripheral it controls.

§3. Modes of transfer

- Data transfer between the CPU and peripheral devices is handled in one of the following 3 possible modes (I/O techniques) : (3.1)
  1. Data transfer under program control or **programmed I/O**
  2. **Interrupt** initiated data transfer(interrupt driven I/O)
  3. Direct Memory Access (**DMA**) transfer

- **Program-controlled transfers** are the result of *I/O instructions* written in the computer program which is in progress. *Each transfer is initiated by an instruction in the program*. As ascertained before, each transfer is between a register of CPU (for instance, the Accumulator) and the peripheral device, or between a register of CPU and the memory. (3.2)

  Transferring data under program control requires *constant monitoring* of the peripheral device by the CPU to identify when a transfer can again be carried out. (3.3)

  In the program-controlled transfer, the *CPU stays in a program loop* until the I/O unit indicates that it is ready. (3.4)

  This is a *time consuming process* since it keeps the CPU busy needlessly. (3.5)

- In case of **interrupt-driven I/O** it is used the interrupt facility of the CPU. When running a program, an I/O instruction informs the I/O unit to issue an **Interrupt Request** when the peripheral device becomes available. After that the CPU switches to another program, while the I/O unit watches the peripheral device. When the device is ready for data transfer, it generates an interrupt request to the CPU. Upon detecting this request the CPU stops momentarily the task it is doing, *branches to a service routine* to process the data transfer and then returns to the task it was performing. (3.6)

- In **DMA**, the dedicated I/O unit, called **DMA Controller,** transfers data into or from the Memory **U**nit through the memory bus. The CPU initiates this type of transfer by supplying the starting memory address and the number of words to be transferred and then frees the bus by rising its terminals in (Z) state. *The DMA controller takes control of the bus* and realizes the transfer. After that it issues an interrupt request to CPU to indicate that the transfer ended. The DMA controller frees the bus and CPU regains control of the bus. There are possible other more sophisticated mechanisms of DMA, known as *stealing a memory cycle from the CPU.* (3.7)

§4. I/O Processors (channels)

- It was seen that I/O operations are *time-consuming*; in the classical von Neumann organization these operations reduce drastically the throughput of any CPU (4.1)
- It would be useful if the *overlapping of the CPU activity with I/O units activity* would have been possible, in order to parallelize their activities. (4.2)

- By natural evolution of the I/O units, they were transformed from simple controllers into **I/O processors** or **I/O channels**. (4.3)

- The I/O unit was enhanced to become an I/O processor in its own right, with a specialized instruction set tailored for I/O operations. The CPU directs the I/O processor to execute an **I/O program** that is placed in memory. (4.4)

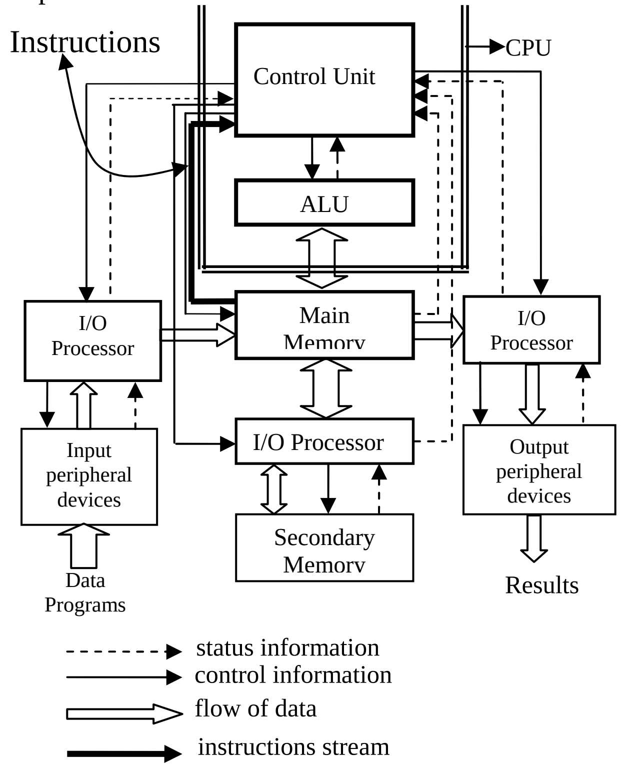
- The I/O processor fetches and executes these instructions *without CPU intervention*. This allows the CPU to continue its *main task*, that of making computations and not manipulating a slow I/O operation. (4.5)

- CPU has to specify to the I/O processor what *kind of I/O operation* is to be run and after that it *will be interrupted only when the entire sequence has been performed*. (4.6)

- I/O processor has its own local memory, its own ALU and control unit. (4.7)

- I/O processor can control a large set of I/O devices with *minimal CPU involvement*. (4.8)

- The already presented general structure of a digital computer can be enhanced by including I/O processors : (4.9)



- Thus, CPU is *relieved* of I/O related tasks and improves its performance. (4.10)

- *I/O processor has the ability to execute I/O instructions* which gives it the complete control over I/O operations. CPU initiates an I/O transfer by instructing the I/O processor to execute a certain program which specifies the device involved in operation, the area of memory for storage, priority, and actions to be undertaken for error conditions situation. The I/O processor follows these instructions and controls the data transfer to/from peripheral device.
- Computer systems are containing a **hierarchy of processors**, where the prime role is assigned to the CPU. (4.11)

§5. Selector and multiplexer I/O Processors (Channels)

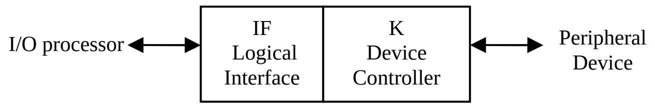
- Two types of I/O processors are known in computer system (5.1)
  - (1) **selector**
  - (2**) multiplexer**

- The **selector channel** controls multiple *high-speed devices* and, at any time is dedicated to the transfer of data with one of those devices.
  Thus, the I/O selector channel selects one device and caries out the data transfer. (5.2)

- A **multiplexer channel** can handle I/O operations with *multiple low speed devices* and *medium speed devices* at the same time,based on time slicing principle. Depending on the peripheral devices speed of operation the following two types of organizing the transfers can be identified: (5.3)

  - ➢ **Byte multiplexer** – when for a time slice *only a byte* (word) is transferred; this is applied in case of very slow peripheral devices. (5.4)
  - ➢ **Block multiplexer** – when for a time slice a *block of bytes* (words) is transferred; this is applied in case of faster peripheral devices. (5.5)

- As it was mentioned, to each I/O processor there are connected several devices through an *I/O controller* having two sections : *logical interface* (IF) and *device controller* (K) : (5.6)



- The *logical interface* is manipulating digital information, whereas the *device controller* is manipulating signals specific to the physical nature of the peripheral device. (5.7)

- The entire I/O architecture consisting of selector channels and multiplexer channels can be represented as in **Annex 6**.

- In general, CPU – Channel (I/O P) communication may take different forms depending on the particular computer system. (5.8)

- Any modern computer system has several channels; a channel performs extensive error detection and correction, data formatting, code conversion etc. The channel can interrupt the CPU under any error condition. (5.9)

- The shared Main Memory stores the CPU and Channel programs and contains a *CPU/Channel Communication area*. This common area is used for passing information between the two processors (CPU and Channel) in form of **messages**. (5.10)
- The messages refer to (5.11)
  - device addresses
  - memory buffer addresses for data transfer
  - types and modes of transfer
  - address of channel program
  - status of channel
- While initiating an I/O transfer, the CPU firstly checks the status of the channel to make sure that the channel is available. It then places the Control Words into the communication area. (5.12)
- The CPU issues the **START I/O** signal to enable the I/O operation. (5.13)
- The channel gathers the *I/O parameters*, executes the appropriate **channel program**, transfers data between devices (if requested) or into/from the Main Memory by acquiring the memory bus (using bus arbitration protocol). (5.14)
- The CPU can continue its activity with another program while the channel (I/O P) is busy with the I/O program. (5.15)
- Once the transfer is complete, the channel sends an Interrupt Request to the CPU through **DONE I/O** line (5.16)
- The CPU responds to the interrupt by issuing the instruction to read the *status word* from the channel; the channel responds to the instruction by placing the contents of its status report into the memory location specified by the memory address field in the instruction. (5.17)
- The status word indicates whether the transfer has been completed by the channel or if any errors occurred during the transfer. (5.18)