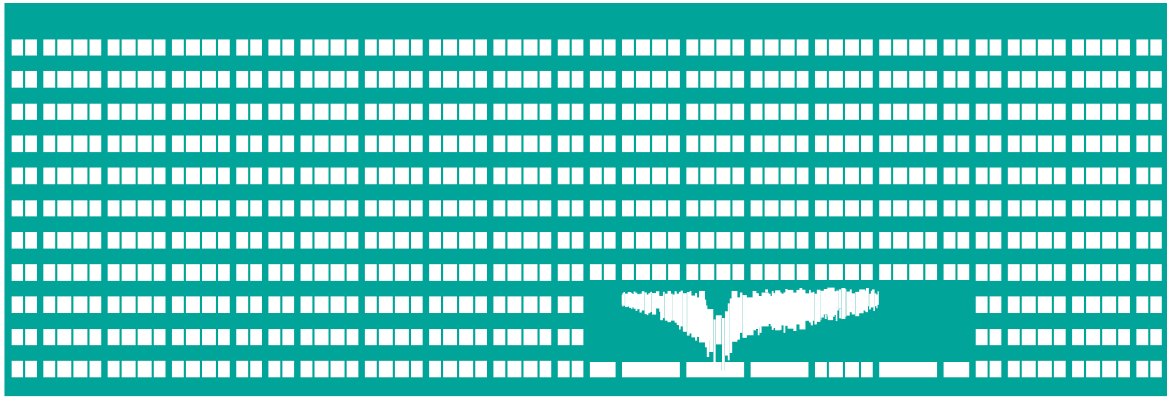
VŠB TECHNICKÁ UNIVERZITA OSTRAVA

VSB TECHNICAL UNIVERSITY OF OSTRAVA



www.vsb.cz

Architektury počítačů a paralelních systémů / Architecture of Computers and Parallel Systems

Part 02: Communication with Devices

Ing. Petr Olivka, Ph.D.
Department of Computer Science,
FEECS, VSB-TUO
petr.olivka@vsb.cz
http://poli.cs.vsb.cz

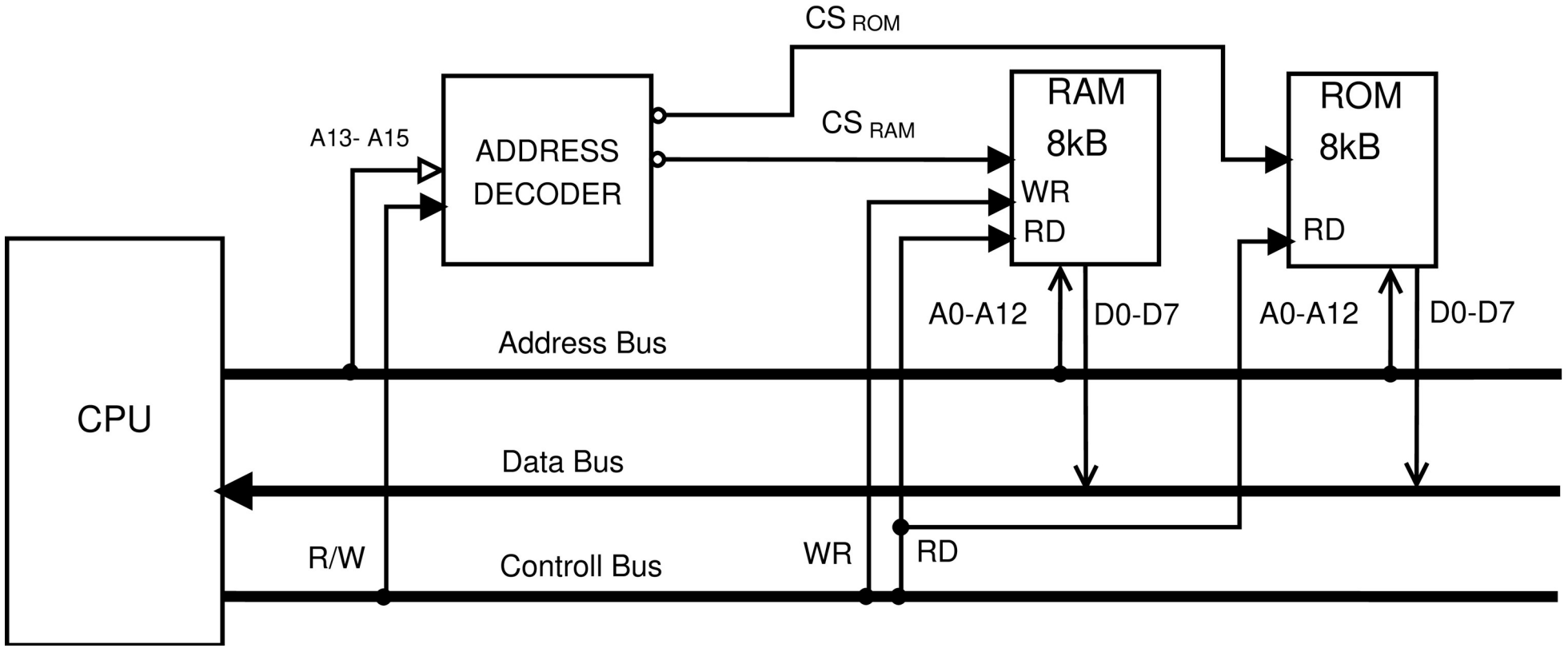
BUS (Address, Data, Control)

This chapter is focused on communication between CPU and devices. According to von Neumann architecture, all parts of computers are connected together by bus. Bus is a bundle of parallel wires separated to three main parts:

- Data Bus – wires are marked as D0 ÷ DN, where N is the number of data bits (wires) used for transmission.
- Address Bus – wires are marked A0 ÷ AM, where M is the number of bits used for addressing.
- Control Bus – set of control signals to control activity on bus.
- Reset – signal is used to initialize all devices connected to bus.
- RD/WR – control direction of data transmission from/to devices.
- MEMR/MEMW – control data transfer from/to memory.
- Clock, Interrupt, DMA, Power Supply and other as specified.

Memory Connected to BUS

Very simple example of how RAM and ROM can be memories connected to CPU by Bus. There is 16 bits CPU with 16 bits Address Bus and 8 bits Data Bus. The way it works is going to be explained on next slide.



... Memory Connected to BUS

Now let's go to how it works. CPU uses 16 bits for addressing. It allows to address up to 64kB of memory. The chip of RAM with capacity 8kB and the chip of ROM with 8kB are connected to the bus:

- To address 8kB, it is necessary to use 13 bits (2^{13}=8192), thus signals A0÷A12 are connected directly to RAM and ROM chip.
- The signal RD and WR (ROM is read only and thus WR is not connected) from control bus are used to control direction of communication.
- Address decoder generates signal CS (Chip Select – signal for chip activation) for RAM and ROM from three highest bits A13÷A15. These 3 bits allow 8 combinations, which will be used depending on computer design. For example bits 000 will generate CSRAM and 111 activate CSROM. All other combinations can be ignored.
- Data from/to RAM and ROM are transferred by data bus D0÷D7.

... Memory Connected to BUS, Multiplexing

The computer is like an easy construction set as can be seen on previous diagram. We only need to connect the right signals from the bus to the input and output pins of the chip.

Individual wires on Data and Address Bus are equivalent form of binary value, because all signals work only on two levels – 0 & 1.

In some computers, where low-cost is important, it is possible to reduce the number of wires in the bus. Some parts of bus in this case are **multiplexed**. This means, e.g. signals A0÷A7 are shared with D0÷D7. In the first step the signals are sent to Address Bus and in second step the same wires are used to transfer data.

Multiplexing reduces computer's speed, but it makes it cheaper and easier. For many applications, where speed is not critical, the multiplexing is a good solution.

Address Decoder, Three-State Chip Output

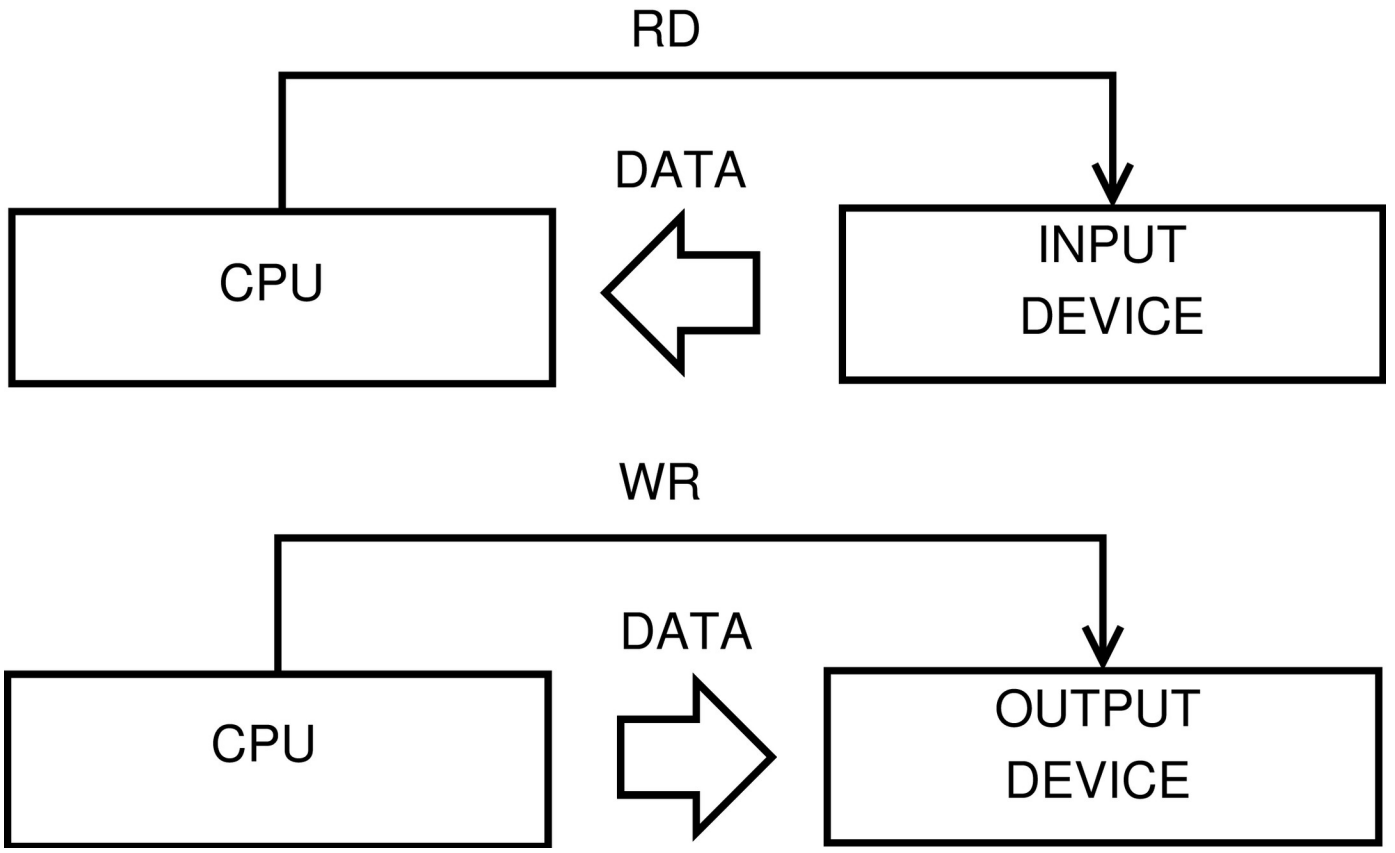
In previous example we used **Address Decoder**. The term seems to be very technical, but it is a very simple circuit. Actually, the Address Decoder is a **comparator** of input value given by signals AM÷AN and **stored value**. When both values are the same, Address Decoder activates the output pin. Address Decoder can be connected to all signals on the Address Bus, or can use only selected signals. It depends on the computer design.

All chips connected to a Data Bus must satisfy the requirement that pins on chip connected to signals D0÷DN are **three states**.

When chip reads data from the Data Bus, pins must be in **input mode**. In situation, when chip writes data to the Data Bus, its pins must be in **output mode**. And the third state is when chip is not active. All pins connected to Data Bus are in **high impedance** state **not to disturb other chips** on the bus.

Communication with Devices, I/O Ports Method

The simplest way for communication with devices is the usage of Input and Output Ports. The device connected to bus has integrated data register, called Port, where data are exchanged between CPU and Device. Next scheme shows the simple scheme for reading and writing data from/to device:



... I/O Ports Method

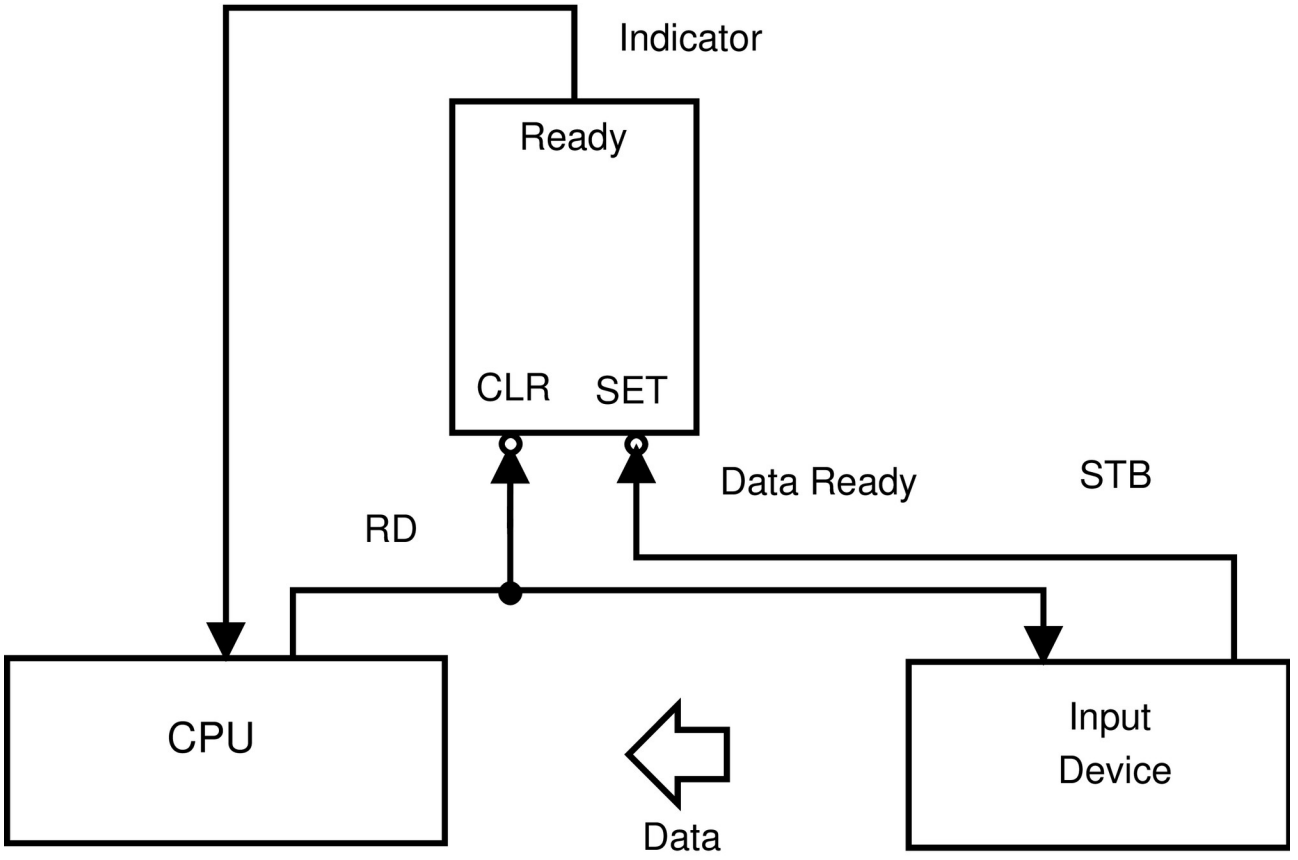
The I/O Ports method is simple. When an executed program requires data from device, it performs instruction for reading from the bus. The Control Unit in CPU generates RD signal and device writes data to the bus. Writing works in the same way, but with the WR signal and data are passed to the bus by CPU and the device accepts them. This method is used very rarely. All devices must be always prepared for reading and writing. This condition can be satisfied only by a few devices.

The devices that are still prepared to pass data to CPU are e.g. devices measuring continuous analog value (temperature, pressure, voltage, ...). Devices prepared to accept data from CPU at any time are not too common. Perhaps only LEDs could be the example.

The main disadvantage of this simple method is that direct usage of I/O Ports does not use any form of feedback and thus data may be lost during transfer.

Communication with Dev., I/O Ports with Indicator

The main disadvantages of raw I/O Ports usage – missing feedback – can be solved by an **Indicator**. It is implemented by RS Flip-flop. The output signal from this circuit is used to let the CPU know that data are prepared. Here is the schema of the input device with Port and Indicator:



... I/O Ports with Indicator

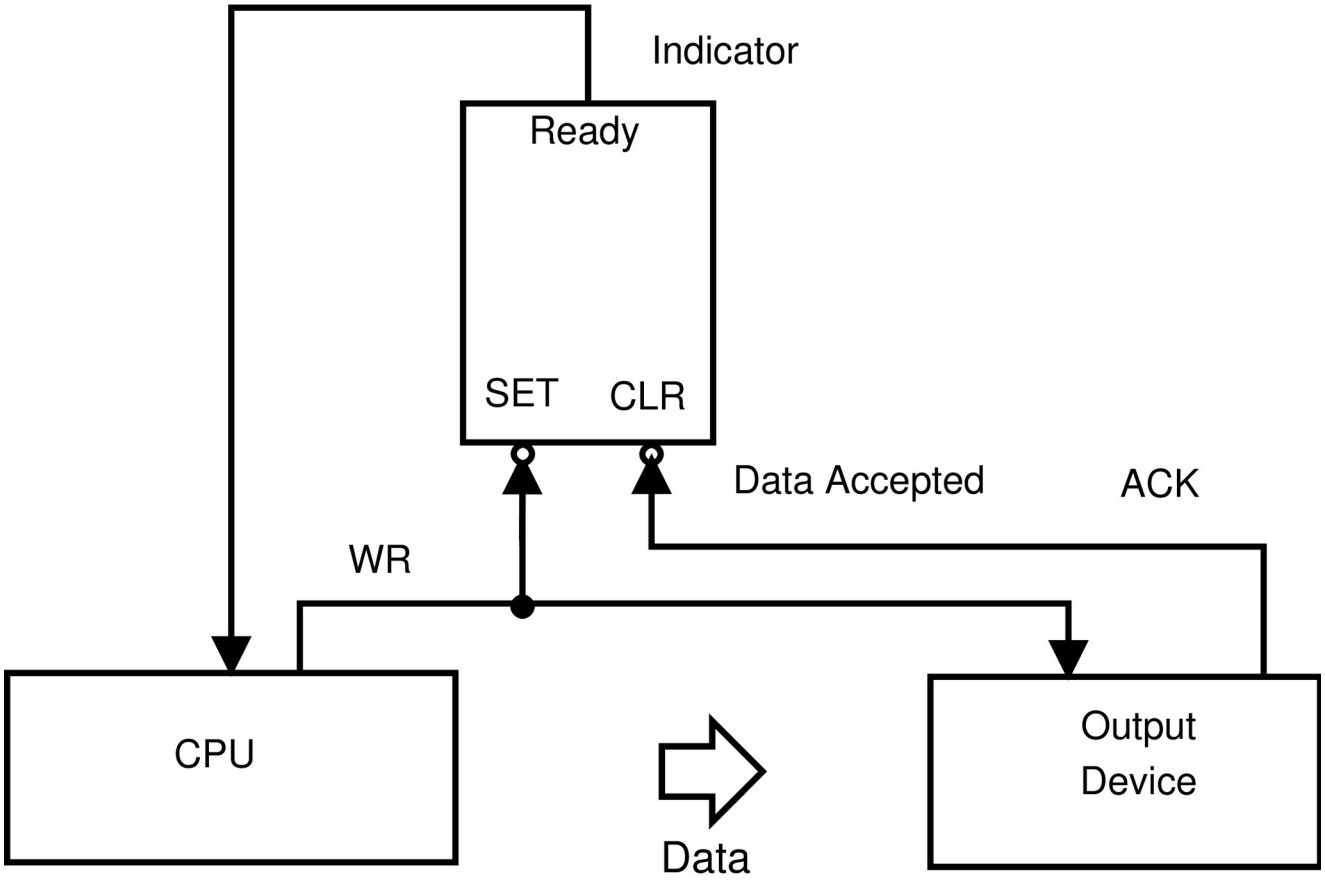
How the Indicator works is clear. When Input Device has the data prepared, it uses STB signal (Strobe) to set the Indicator. Its output is observed by executed program. If indicator is set, it is possible to read data safely. Signal RD will clear the indicator during reading. It remains not set until new data are prepared for reading.

This method will still have a problem with data loss. If CPU delays reading from the device, it may fail or may overflow internal register or buffer.

This method, when the program first checks the indicator in the loop and then reads data, is called **spooling**. It is not too effective, because it consumes too much of CPU's performance.

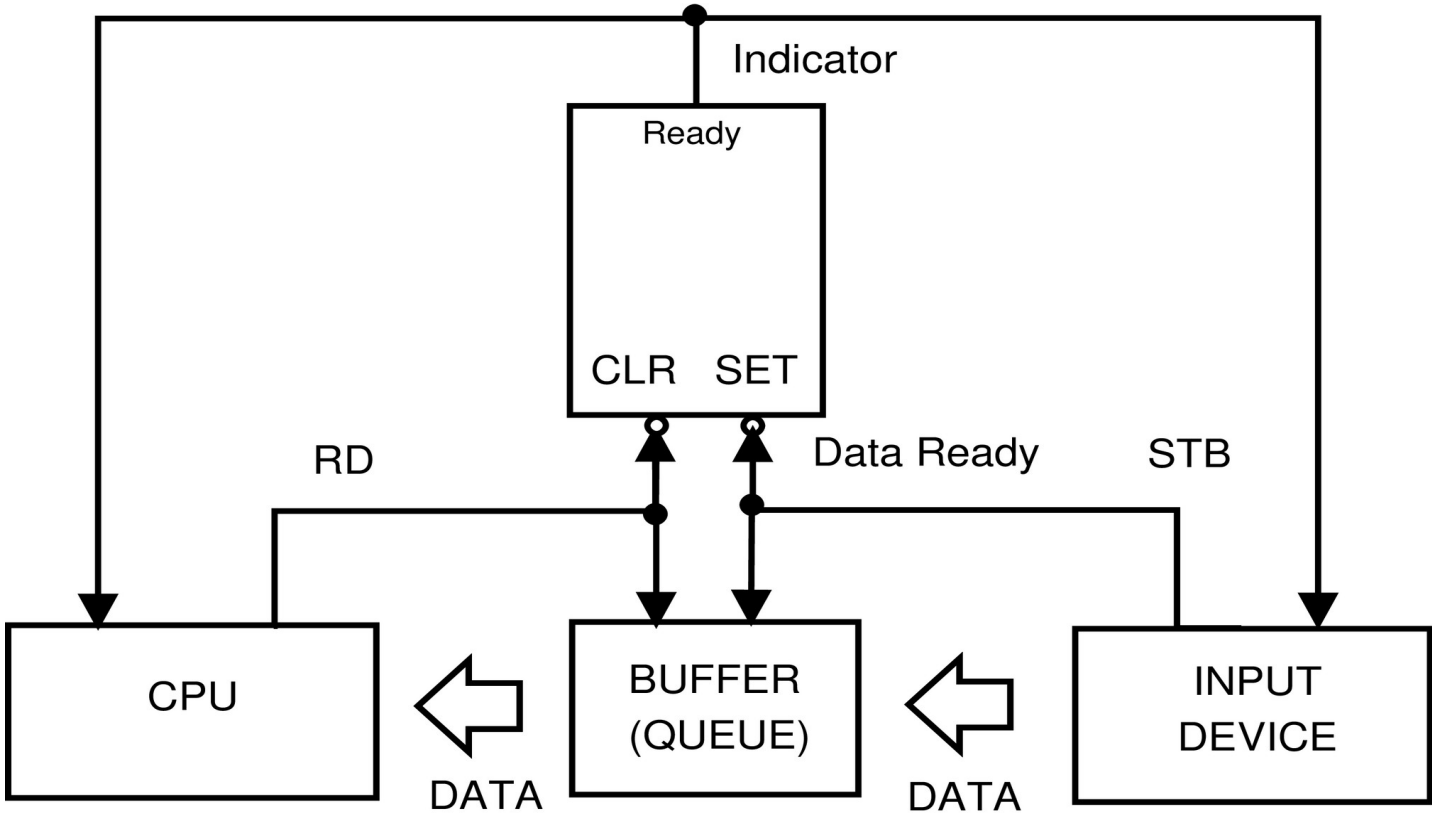
… I/O Ports with Indicator

The Indicator can be used for writing in similar way as for reading. The program in CPU must check the Indicator before it writes data, to make sure, that the previous data was accepted. How it works can be seen on the next scheme:



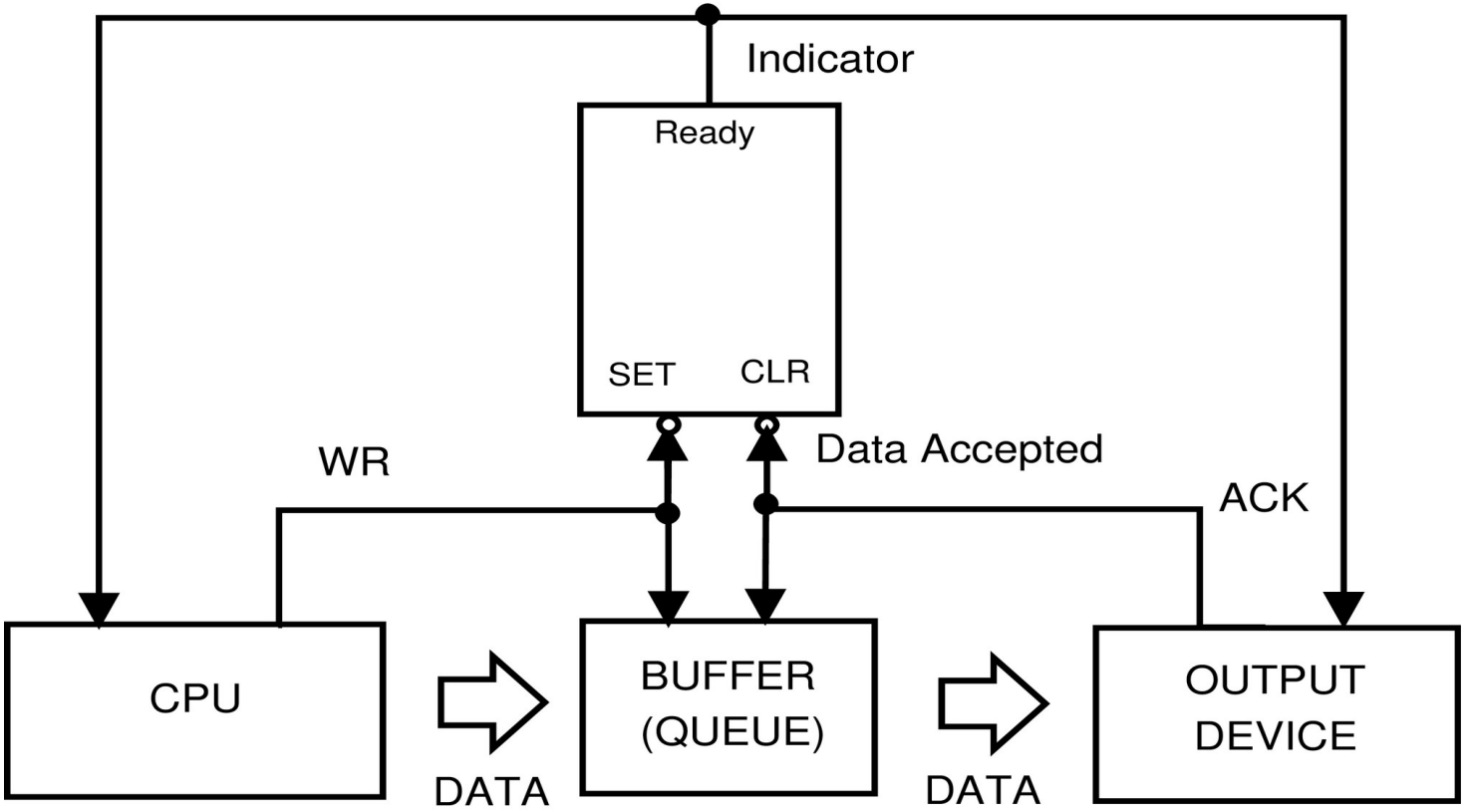
Communication with Devices, Device with Buffer

The usage of the Indicator is insufficient for time critical devices or devices with big data throughput. In this case, it is necessary to implement the Buffer (Queue) between the Device and the CPU. The Indicator is implemented as well and it is used by the CPU and the Device, as is shown on the next scheme:



... Device with Buffer

The Buffer between the CPU and the Device allows to send a block of data in one step. It improves throughput from/to the device. The buffer can be used for reading in the same way as for writing, as it is visible on the next scheme:



Communication with Interrupt

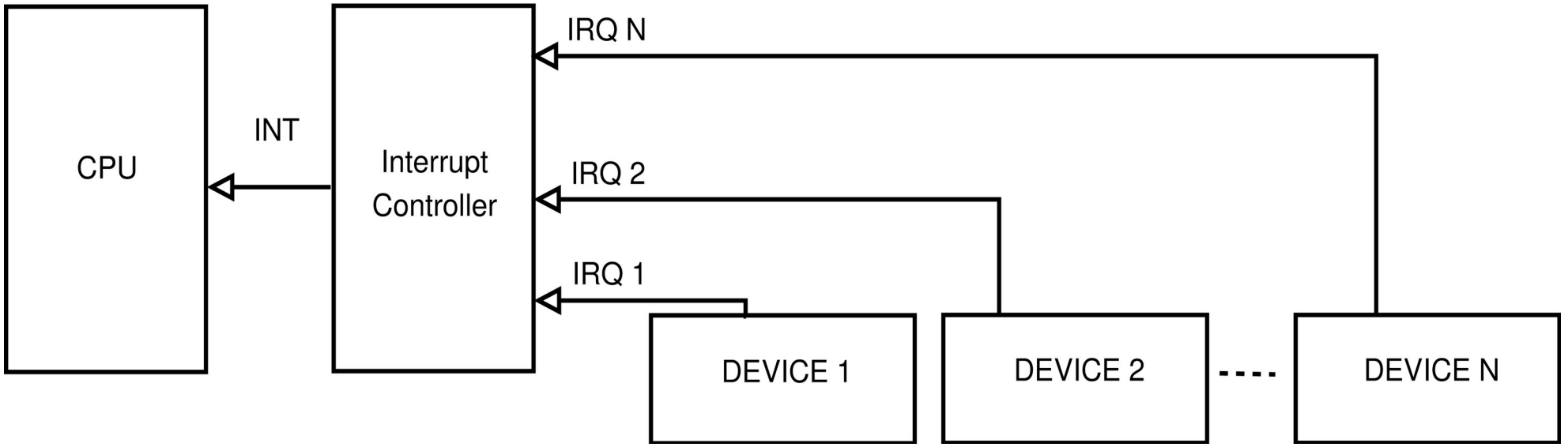
Using the indicator for multiple devices is uncomfortable and overloads CPU (spooling mentioned earlier). Thus CPU manufacturers introduced a better technology for communication with devices – called **interrupt**. It is an event generated by the device, which **interrupts** the execution of the main program, then the CPU calls the interrupt routine and when the device is handled, the CPU returns the execution back to the main program. It is very effective. No useless testing of indicators in the loop is needed.

This technology does not require construction changes for devices. Signal indicator is changed to **IRQ** (interrupt request) and it is not directly connected to the CPU, but to an **interrupt controller**.

This circuit is able to serve more interrupt requests at the time, it masks selected requests, it evaluates priority in a different way and is able to communicate with the CPU and map interrupts to different interrupt routines.

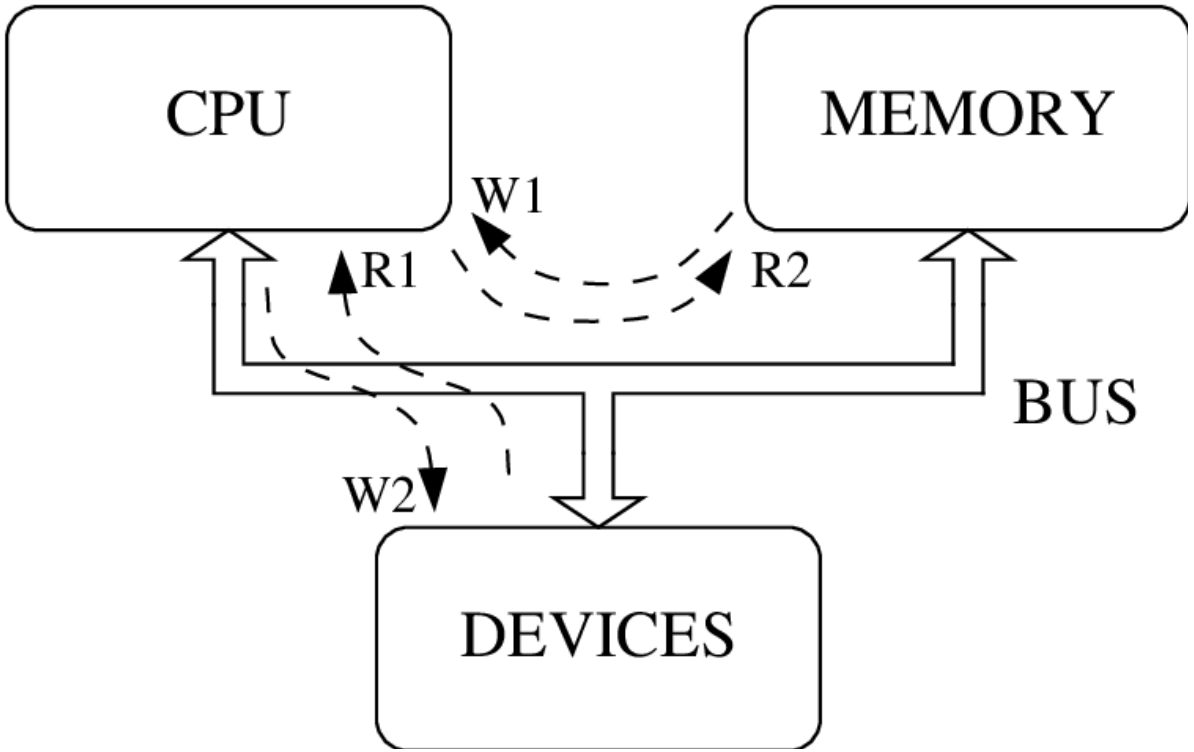
… Communication with Interrupt

How Interrupt Controller collects IRQs from more devices and creates one INT request for CPU, is shown on the next scheme:



I/O Ports are Slow

The communication with devices using I/O Ports has one big weakness. All data must be transported through the bus two times, as shown in the scheme below. When program reads data from the device, the CPU has to read data from the device to the CPU – step R1 and then stores it in the memory – step R2. In the opposite direction, writing data to device needs two steps too – W1 and W2.



DMA – Direct Memory Access

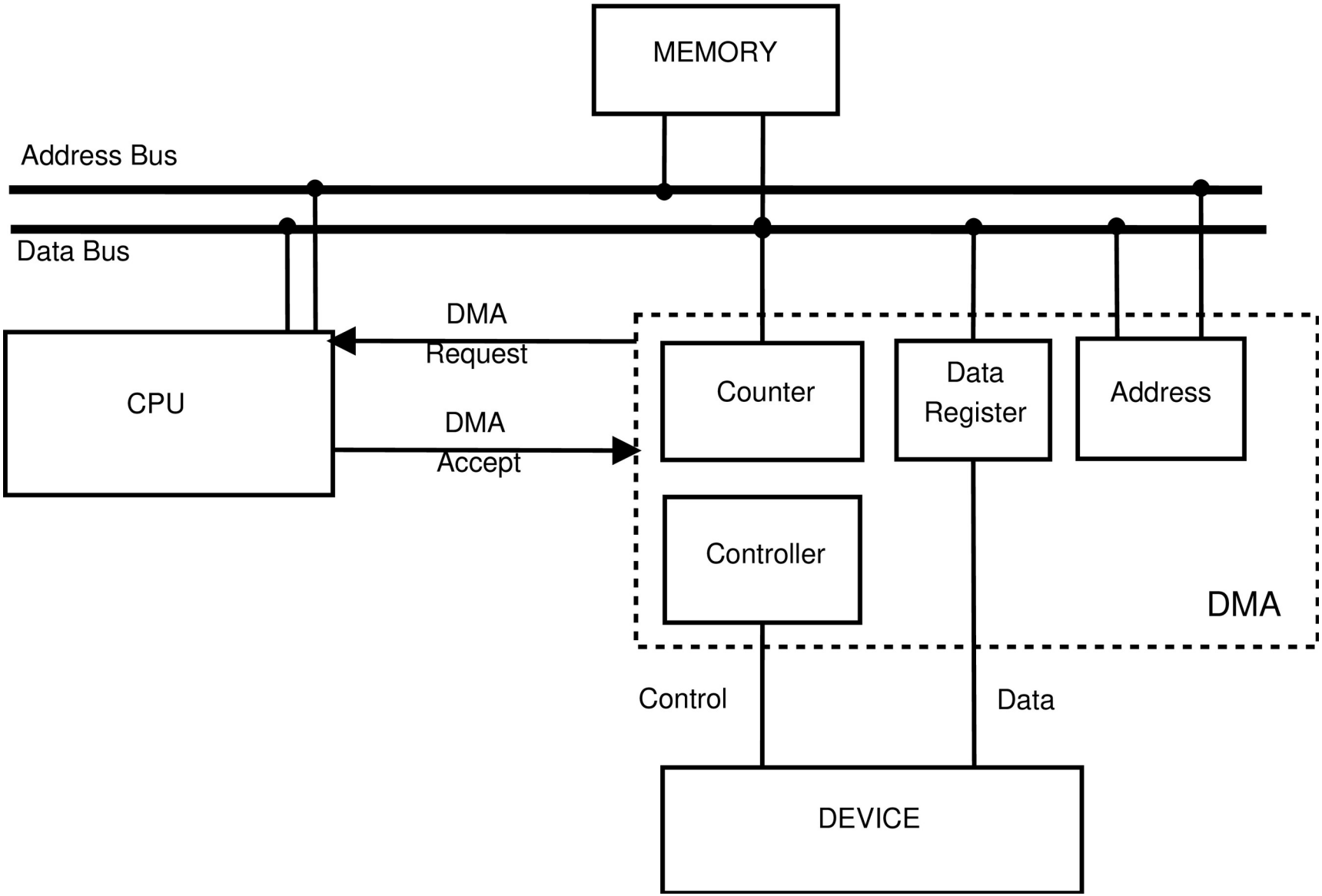
To remove the main problem of I/O Ports – double data transfer - computer manufacturers have designed a new technology: DMA.

This technology improvement made the device a little more sophisticated. All device controllers contain their own bus controller with three registers. The first one for the data transfer (I/O Port), the second one for the address and the third one is the counter. Simple scheme is on the next slide.

The DMA controller controls data transmission from device to memory directly without involving the CPU. This direct data transfer saves expensive CPU time, because the CPU can execute a program and is relieved of slave labor with the data. It also improves the bus bandwidth.

The DMA controller have to cooperate with the CPU, because only one bus controller can control the bus at any moment. The DMA controller and CPU use two signals – DMA Request and DMA Accept.

… DMA – Scheme


MEMORY
Address Bus
Data Bus
DMA
Request
DMA
Accept
CPU
Counter
Data
Register
Address
Controller
DMA
Control
Data
DEVICE

… DMA – Operation Description

Before starting the DMA transfer, the CPU must set the Address Register and the Counter to the initial value. Then it starts transfer for reading data from device (or writing to device):

1. The DMA controller sends DMA-Request to CPU.
2. When the CPU releases the bus, it sends DMA-Accept to allow one DMA transfer.
3. DMA sets value from address register to Address Bus.
4. DMA writes data to Data bus.
5. Memory stores data from bus to given address.
6. DMA increments Address Register and decrements the Counter.
7. One byte is transferred.
8. If the counter is zero, the transmission ends.
9. If the counter is not zero, the DMA continues from point 1.

... DMA and Channels

The DMA is very efficient technology for data transmission and is still used in today's computers. It saves the CPU time required for computing and it uses time, when the CPU does not access the bus for the data transfer. This simple parallelism greatly improves overall system throughput and the computing performance.

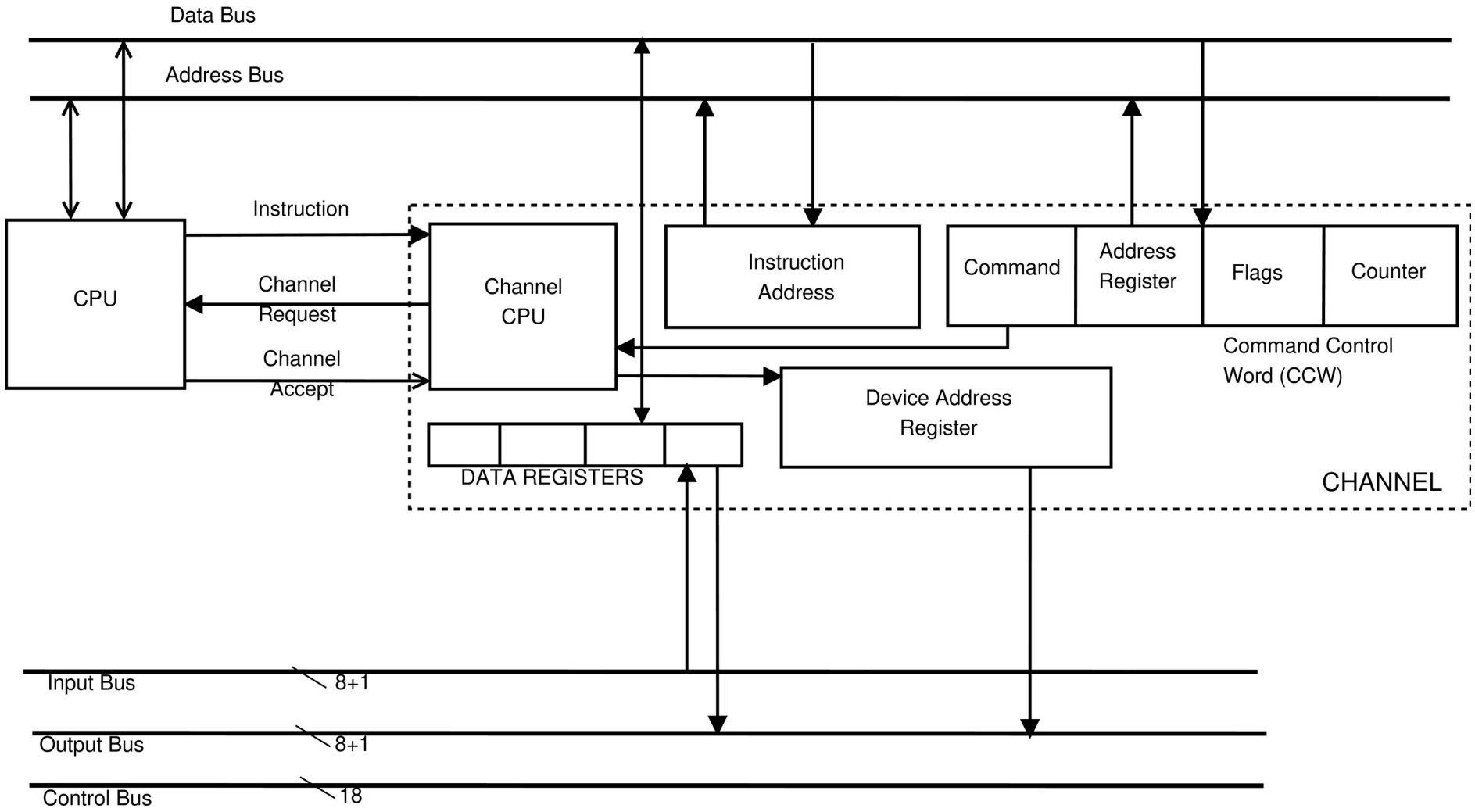
In mainframes the DMA was replaced with **Channel**. It is higher level of the DMA. Channel Controller has its own CPU to directly control the connected devices. The CPU only starts and stops communication.

The one very-well know example of Channel architecture is the SCSI (Small Computer System Interface) technology, introduced in 1981. SCSI-1 was able to transfer up to 5MB/s. Today's latest standard allows transfers of up to 640MB/s.

The successor of SCSI is FC – Fiber Channel. This technology is used for hard disks and disk arrays.

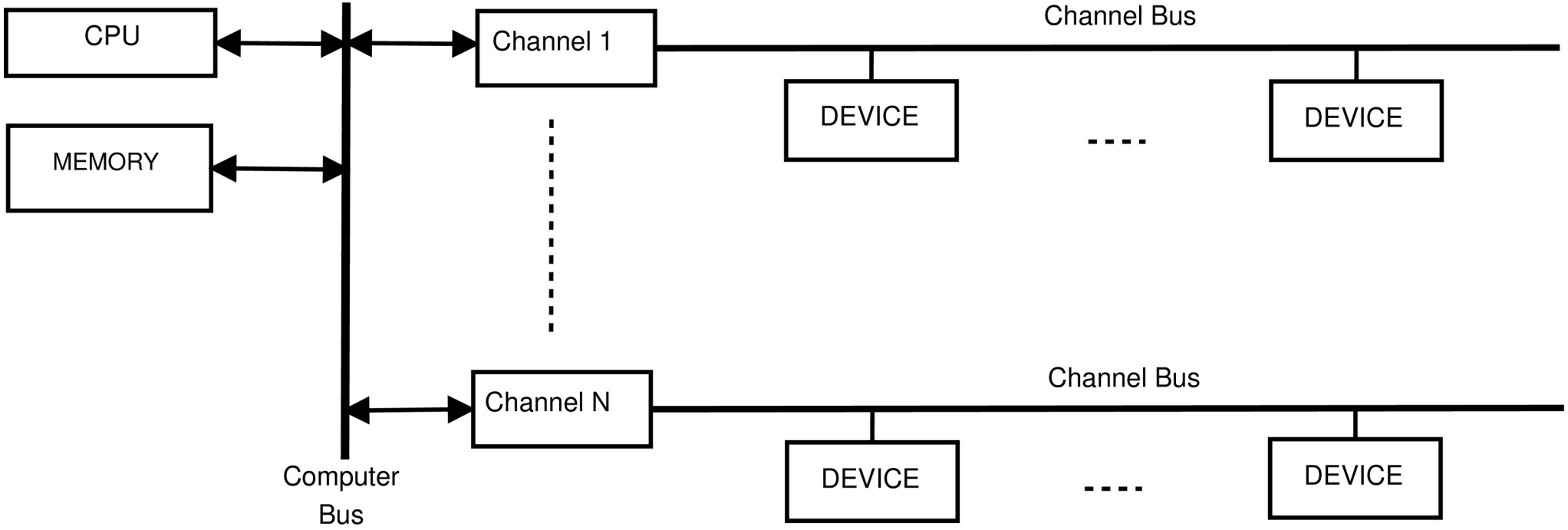
Channel Scheme

The Channel operates similarly to the DMA. It allows to attach more devices.



Multi-Channel Architecture

- Computer can implement more channels to communicate with enormous number of devices, without CPU overloading.



PC Bus History – Table Overview

Name	Year	Bus Width [bits]	Clock [MHz]	Speed [MB/s]
XT Bus	1980	8	4.77	2
ISA	1984	16	8	16
EISA	1987	32	8	do 33
VL-BUS	1992	32	25-66	do 264
PCI	1993	32/64	20-66	132-528
AGP	1997	32	66/133	264-2112
PCI-X	2002	64	66/133	533/1066
PCI-Express	2004	1-128	250	do 32000

VSB TECHNICAL UNIVERSITY OF OSTRAVA
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE
DEPARTMENT OF COMPUTER SCIENCE