I/O

Todd C. Mowry

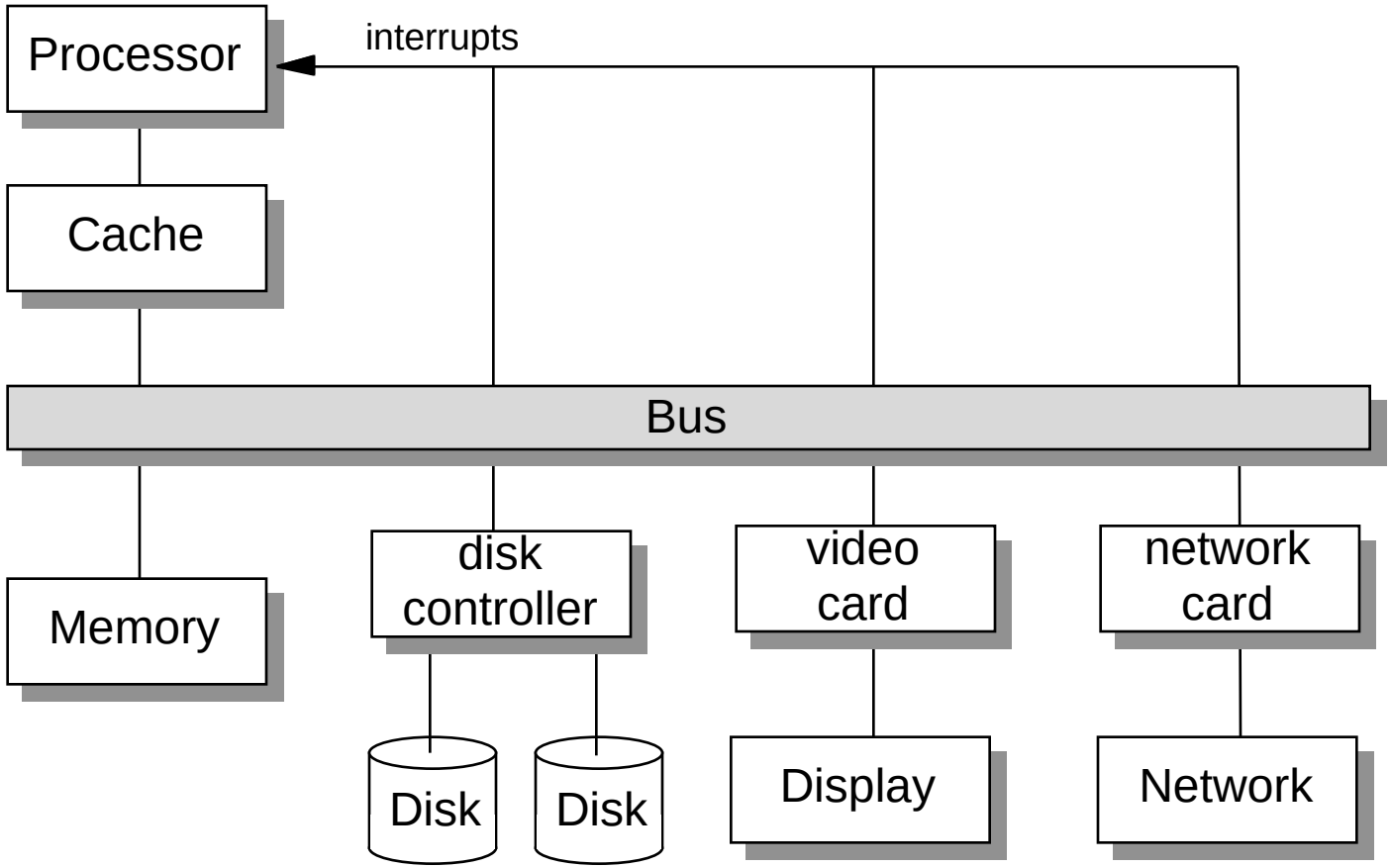
CS347

Feb. 12, 1998

Topics

- buses
- I/O devices
- I/O research at CMU

Computer system

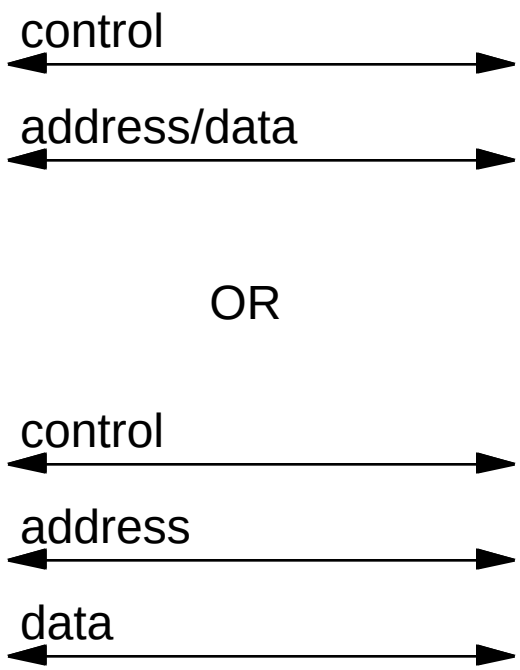


What is a bus?

A bus is a shared medium that connects the processor, memory, and I/O devices

Consists of control and data/ address wires

- **control: requests, acks, type of data (address or data)**
- **data lines: data, addresses**
- **address lines (optional): address**



Bus types

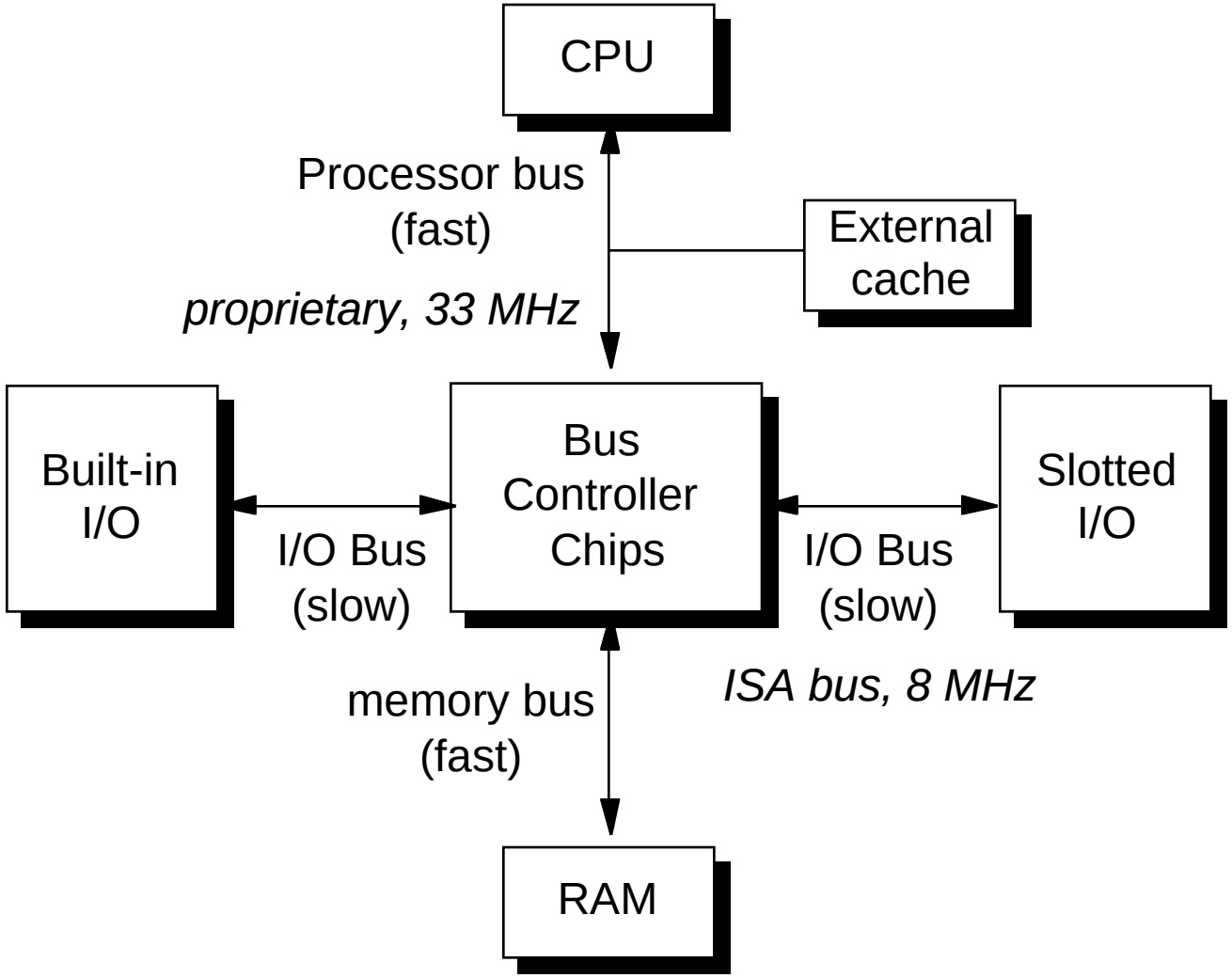
Processor-memory bus

- **short, fast, proprietary**
- **fixed number of devices with known performance**

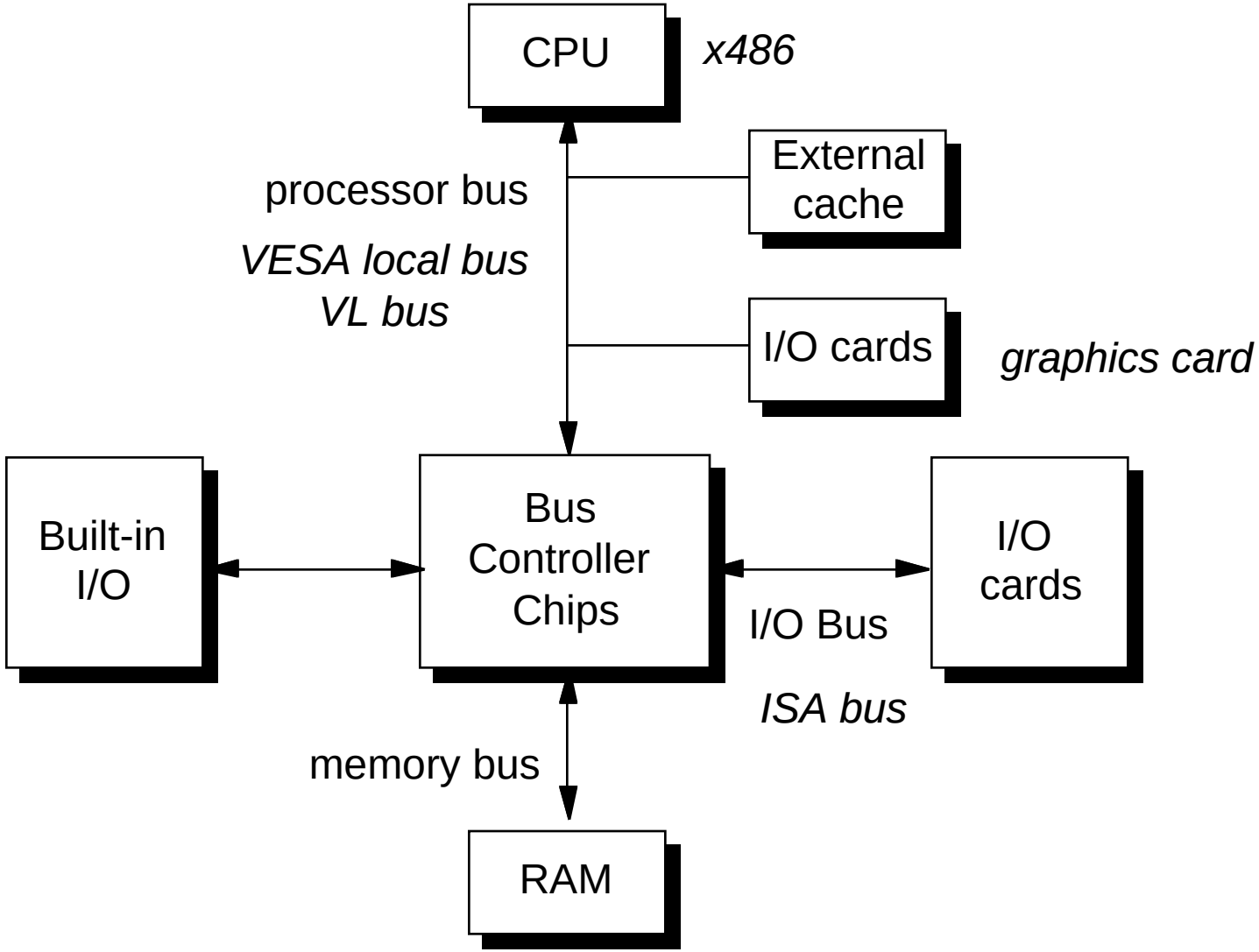
I/O bus

- **longer, slower, open**
- **unknown number of devices with different performance**
  - disk: 5 MB/s
  - 4x CDROM: 640 KB/s
- **Examples: SCSI II, PCI, ISA, EISA**

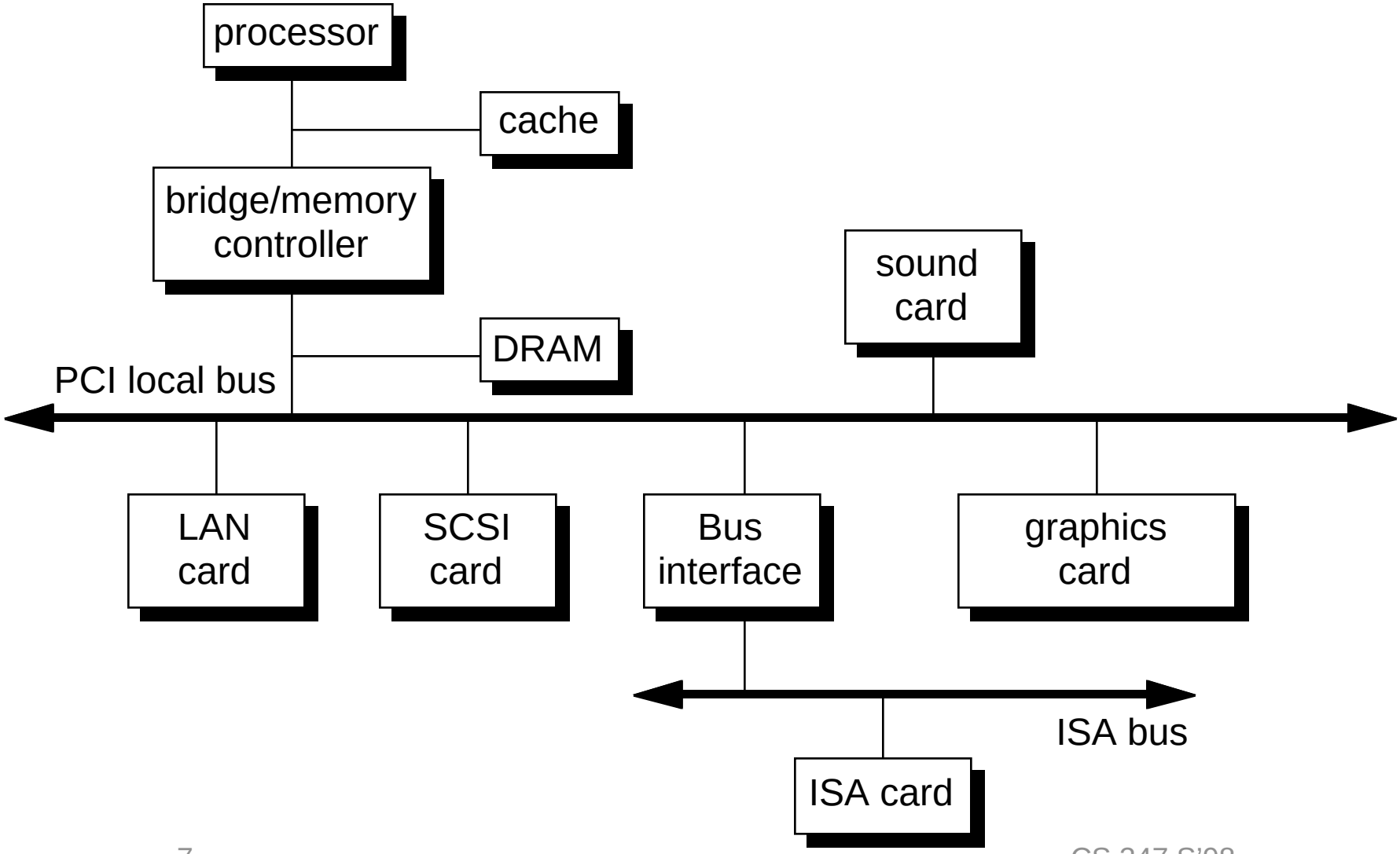
Traditional PC bus layout



Local bus layout



PCI bus layout



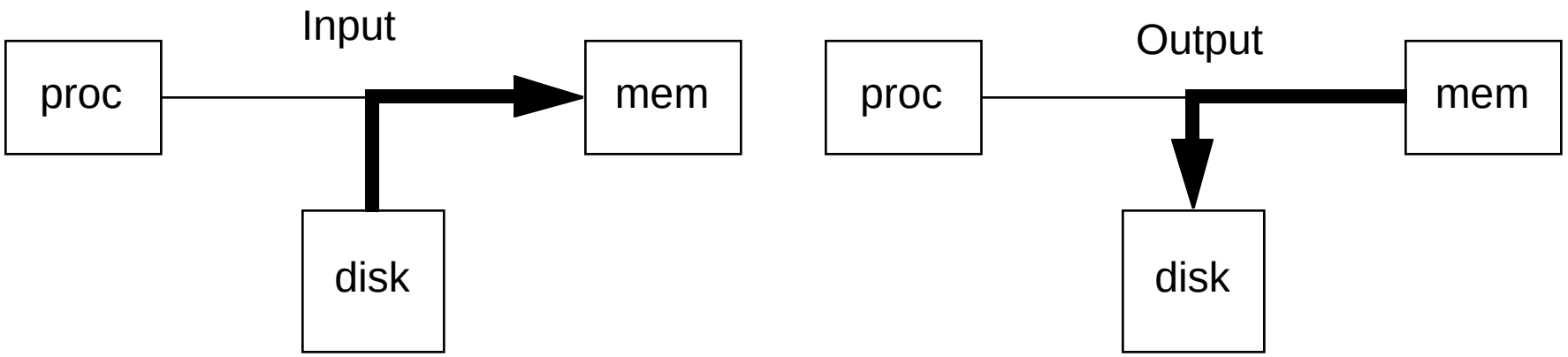
Bus transactions

Input transaction

- **transfers data from some device to memory or processor**

Output transaction

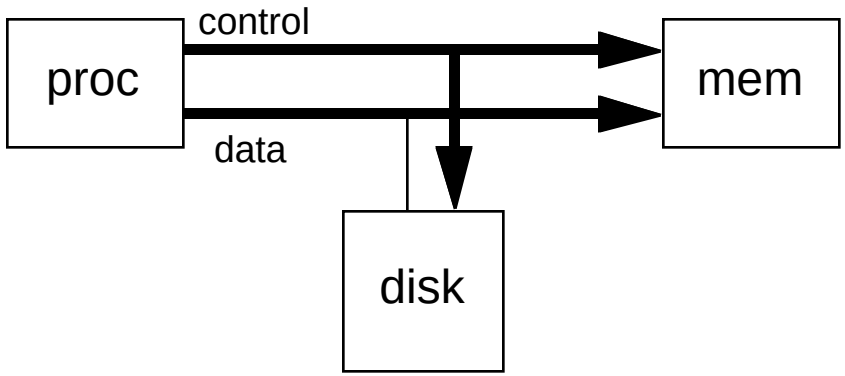
- **transfers data from memory or processor to device**



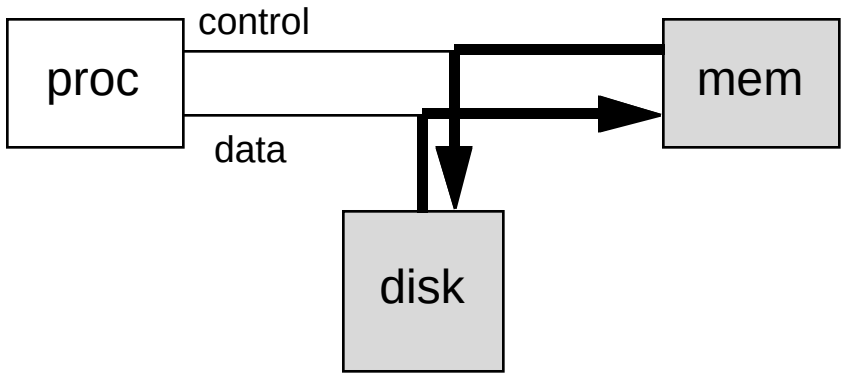
Input transaction

1. Processor requests input on control lines and places memory address on data lines.

(assume disk address handled at a higher level)

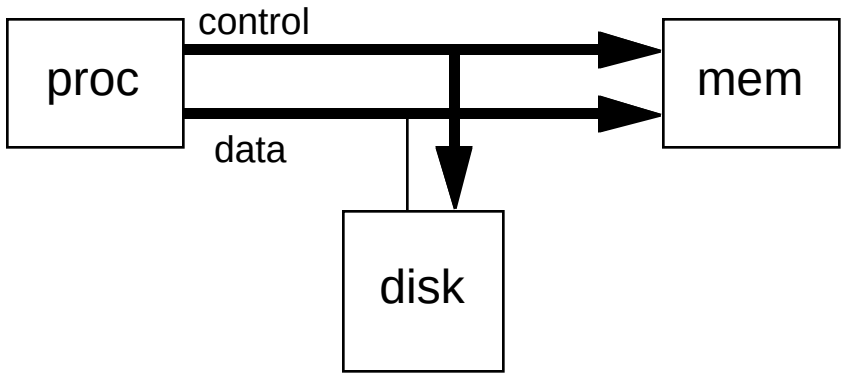


2. When memory is ready, it signals disk on control lines. Disk then transfers data.

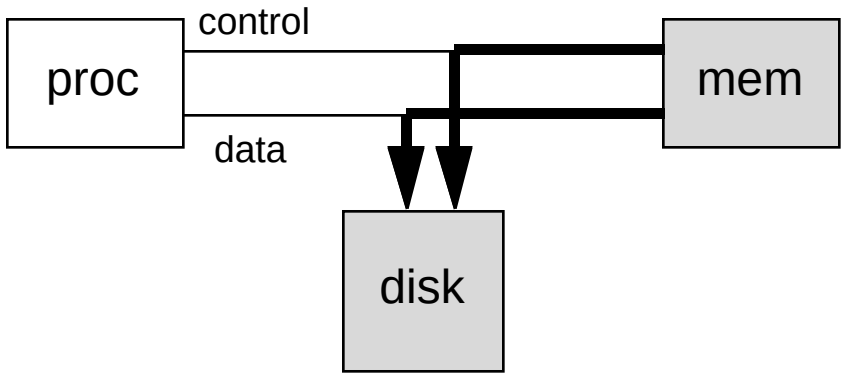


Output transaction

1. Processor requests output on control lines and places memory address on data lines.

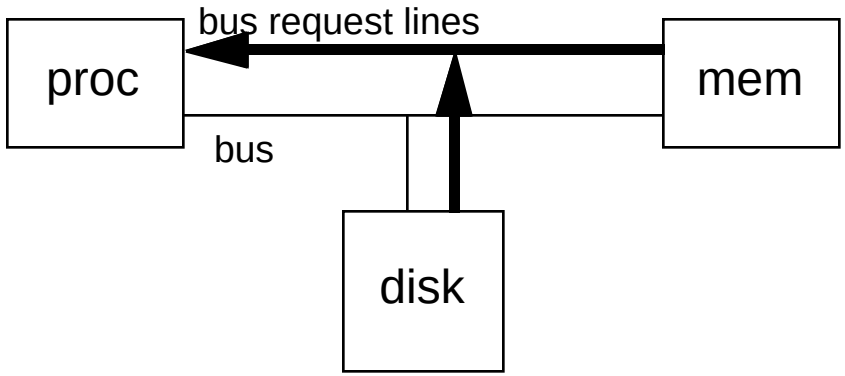


2. When memory is ready, it places data on data lines, indicating availability with the control lines.

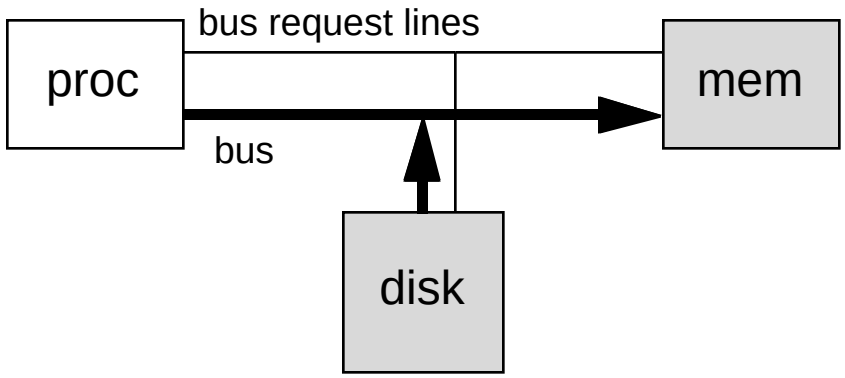


Bus arbitration (one master)

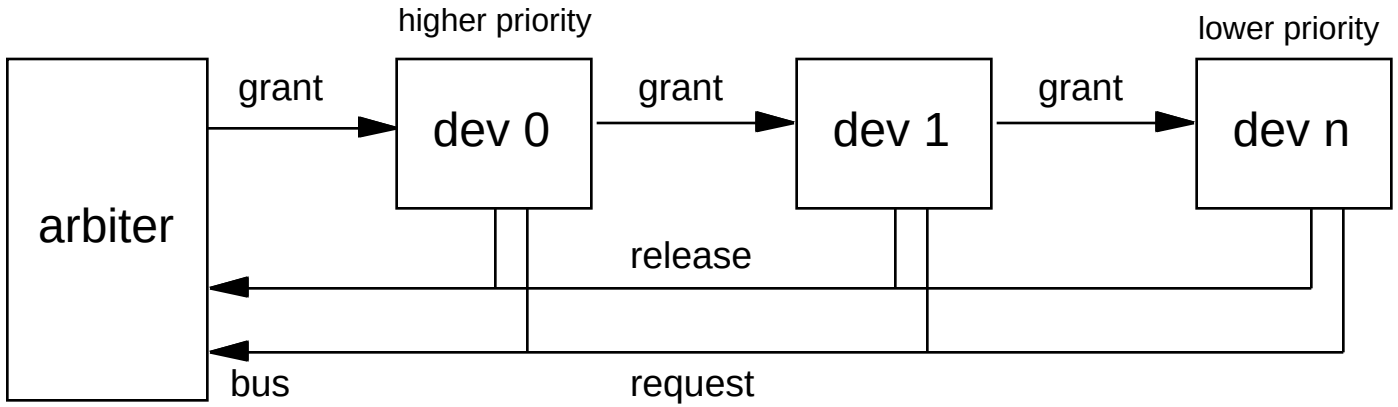
1. Device asks processor for permission to use the bus for an input transaction.



2. Processor initiates transaction on behalf of device.



Daisy-chained arbitration



Not fair!

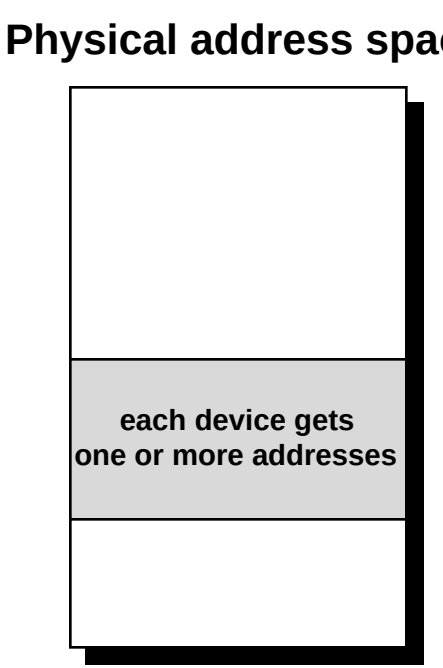
How processors send I/O commands

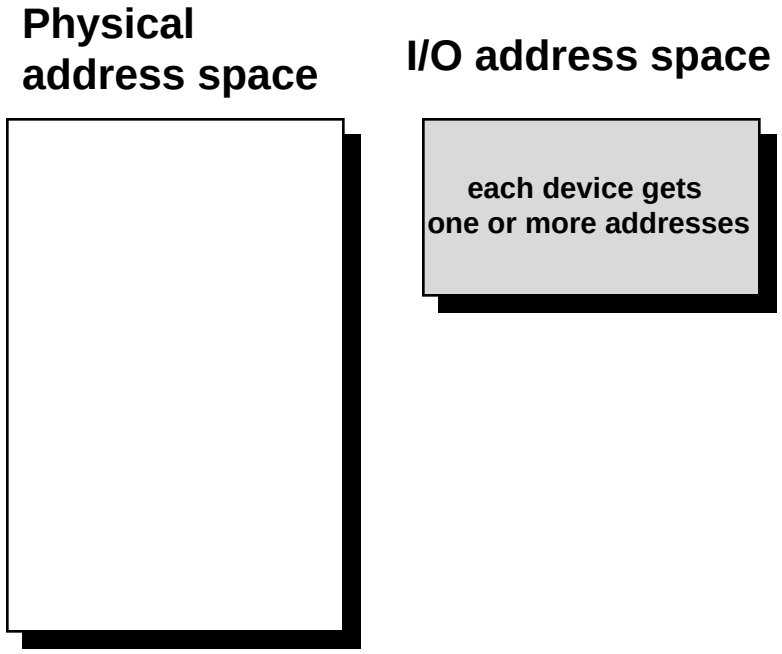
Memory mapped I/O

- special areas of the address space called I/O registers (e.g. MIPS)
- st $2, disk_ctrl_reg -- set disk addr
- st $3, disk_data_reg -- send word
- ld $4, disk_stat_reg -- get status

Explicit I/O instructions

- separate I/O address space (PC)
- in $4, keyboard_reg -- read keybrd char





How processors get I/O status

via polling a memory mapped status register (polled I/O):

```
while (status_reg != READY)  /* spin on a memory mapped status register */
    ;
```

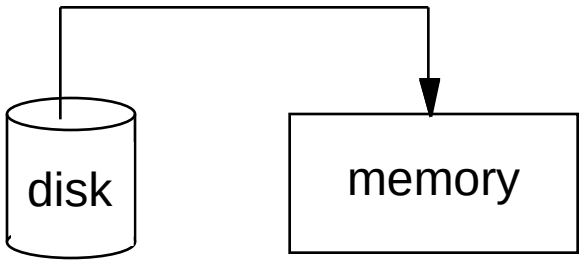
via an exception handler (interrupt-driven I/O):

- I/O device initiates external exception (e.g., when data is ready)
- control branches to handler, which reads status and takes appropriate action (i.e., read data from device)

How data is transferred

Programmed I/O:

```
for (i=0; i<n; i++) {
    while (disk_stat_reg == NOTREADY) ;
    a[i] = disk_data_reg;
}
```



read from disk write to memory

DMA (direct memory access):

- programmable DMA controller serves as bus master

```
st $2 dma_stat_register   -- setup start address and count
st $3, dma_start_register  -- start the transfer in the background
```

PC I/O space

Separate I/O space with 64K addresses (ports)

- **each device assigned one or more ports (for control & data)**
- **bus control line distinguishes I/O address from regular addresses**

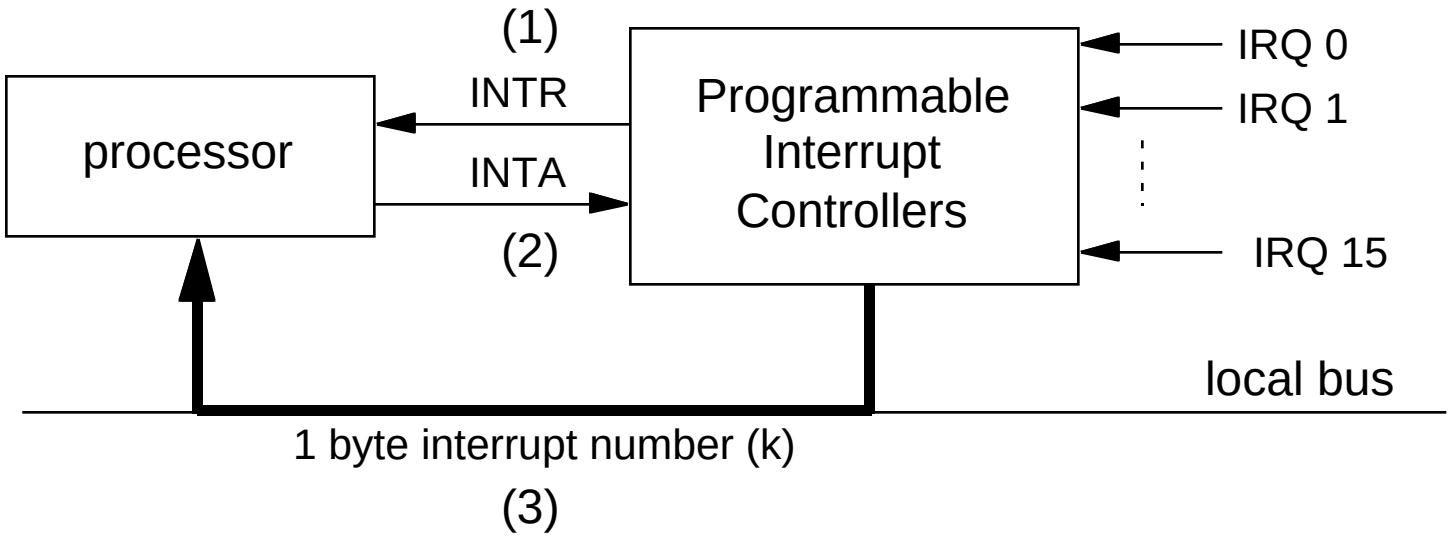
Separate instructions read and write to ports

- **in $1, 0x60 -- read contents of keyboard data register**

Some example ports:

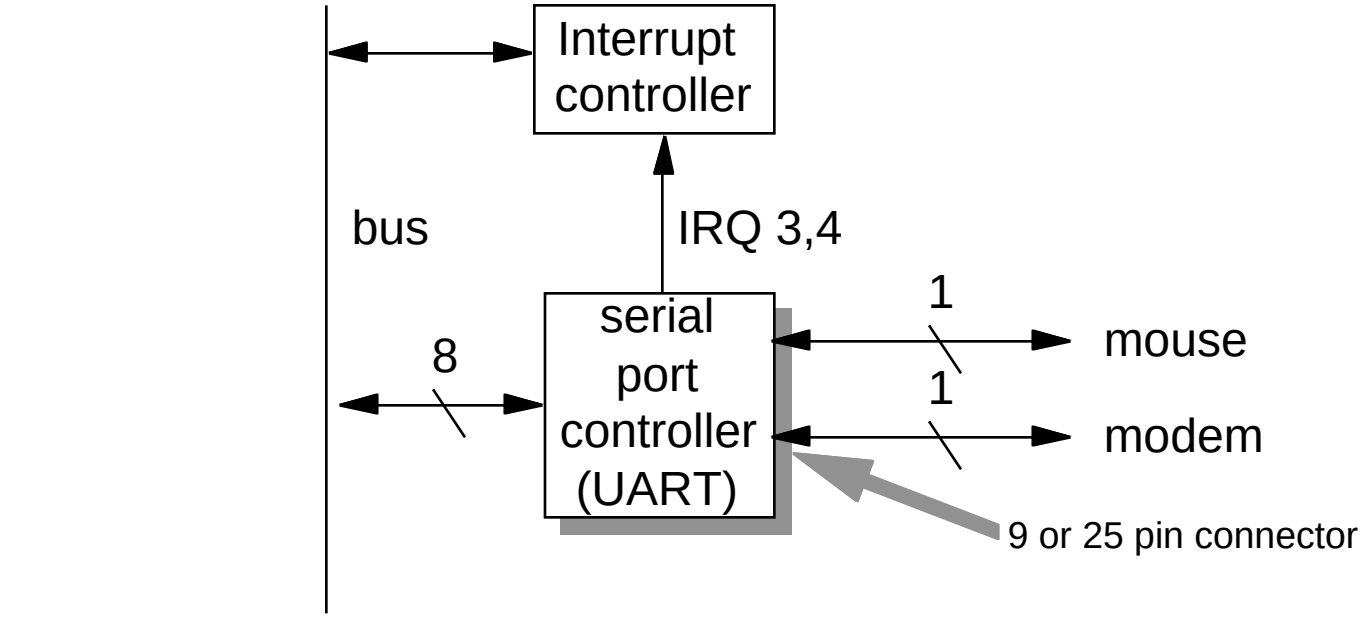
0x60-0x6f	Keyboard controller
0x1f0-0x1f8	Hard disk controller
0x3bc-0x3f	Parallel port controller
0x3c0-0x3cf	Video adapter
0x3f8-0x3ff	Serial port controller

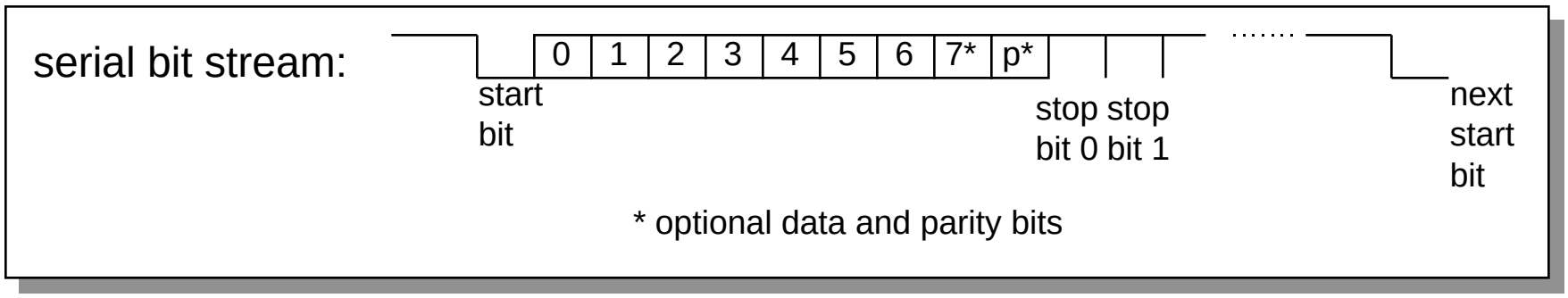
PC interrupts



(4) Take exception to address k*4

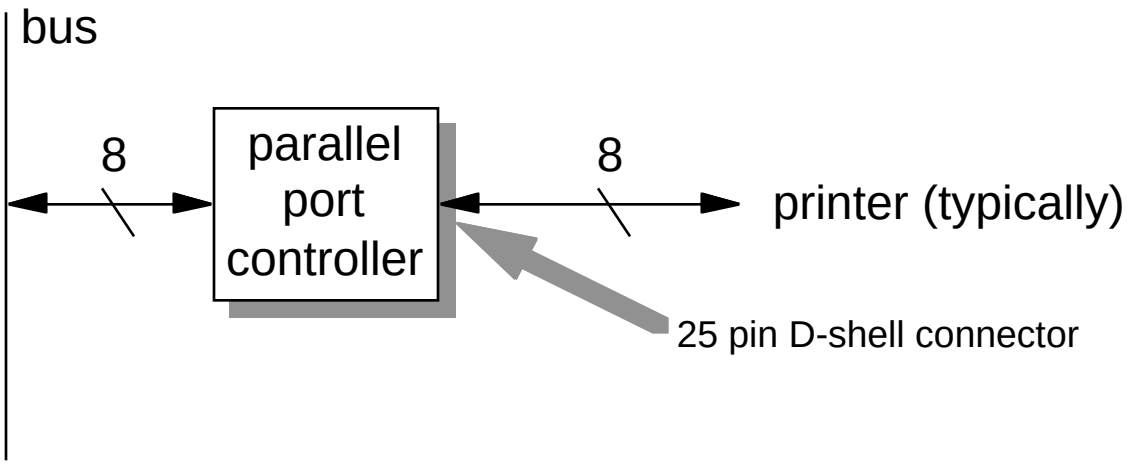
Serial ports





slow (10-20 Kbits/sec transfer rate) but long and flexible

Parallel ports

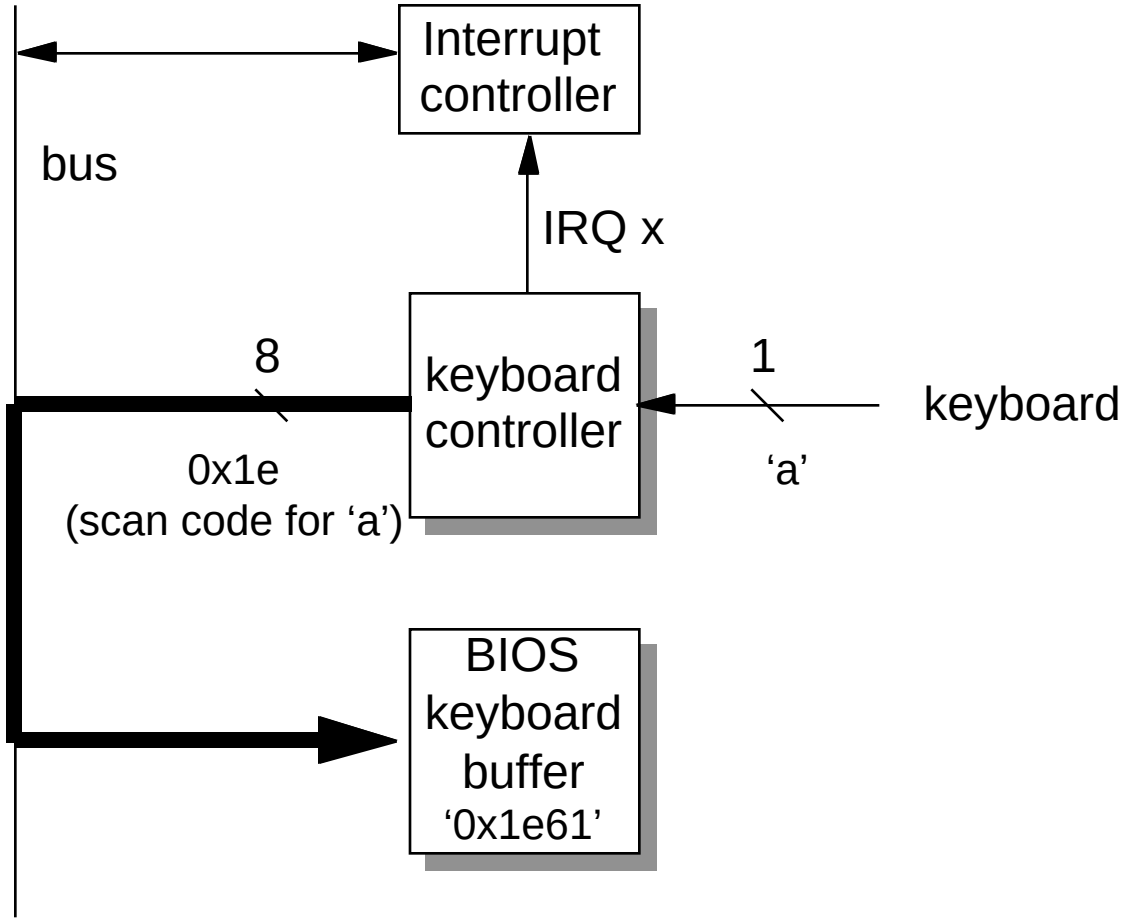


Each parallel port has three I/O ports:

0x3bc	data (read/write)
0x3bd	status (read only)
0x3be	control (poll or interrupt)

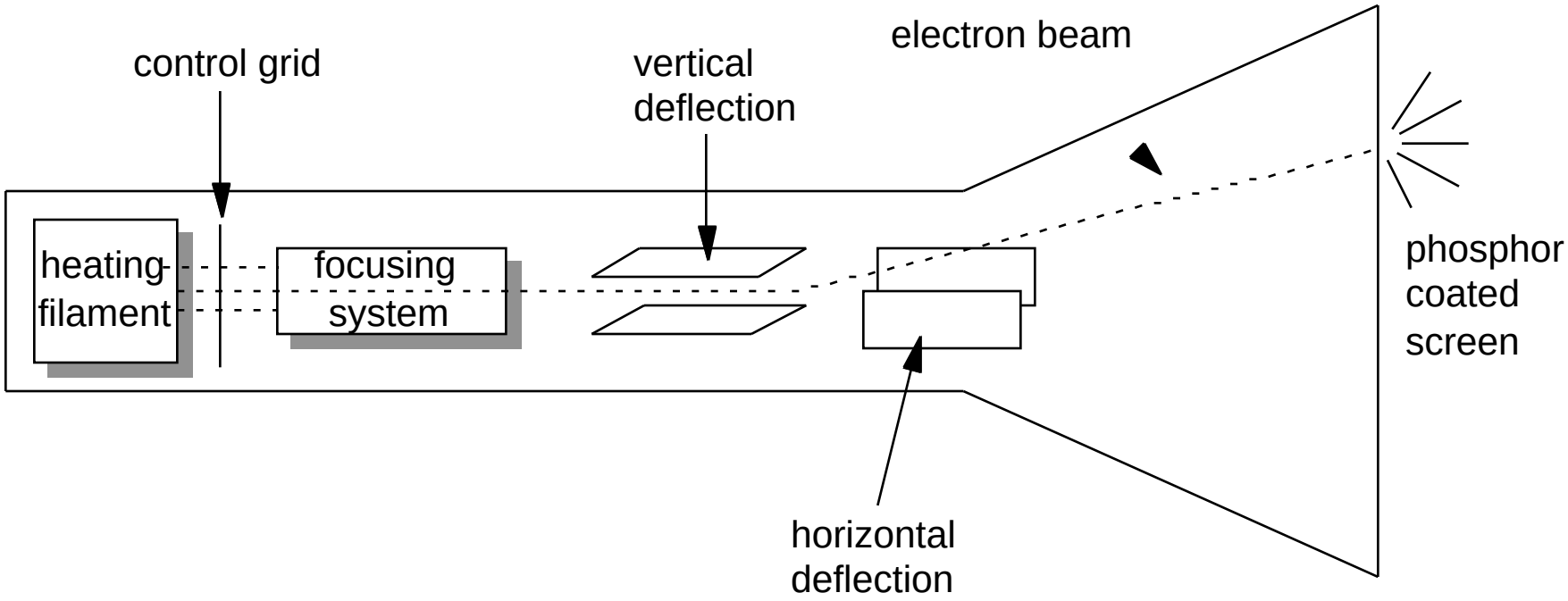
faster, but shorter than serial ports

Keyboard

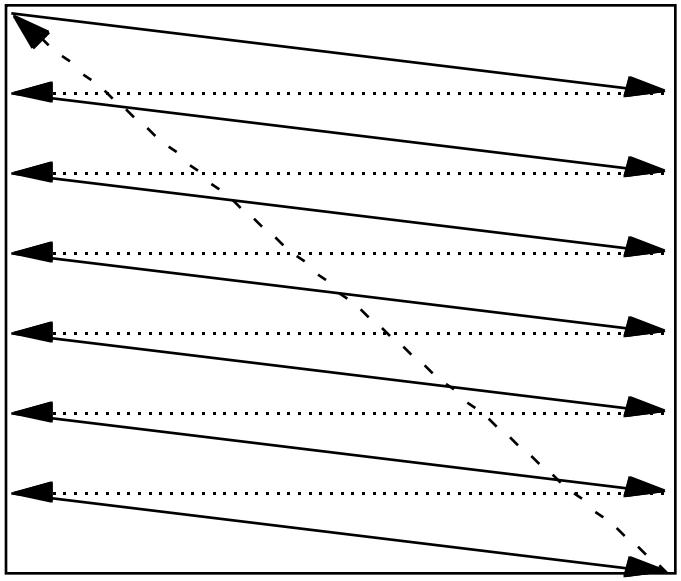


interrupt handler transfers data from keyboard controller to BIOS keyboard buffer

Display


control grid
vertical
deflection
electron beam
heating
filament
focusing
system
phosphor
coated
screen
horizontal
deflection

Raster scan



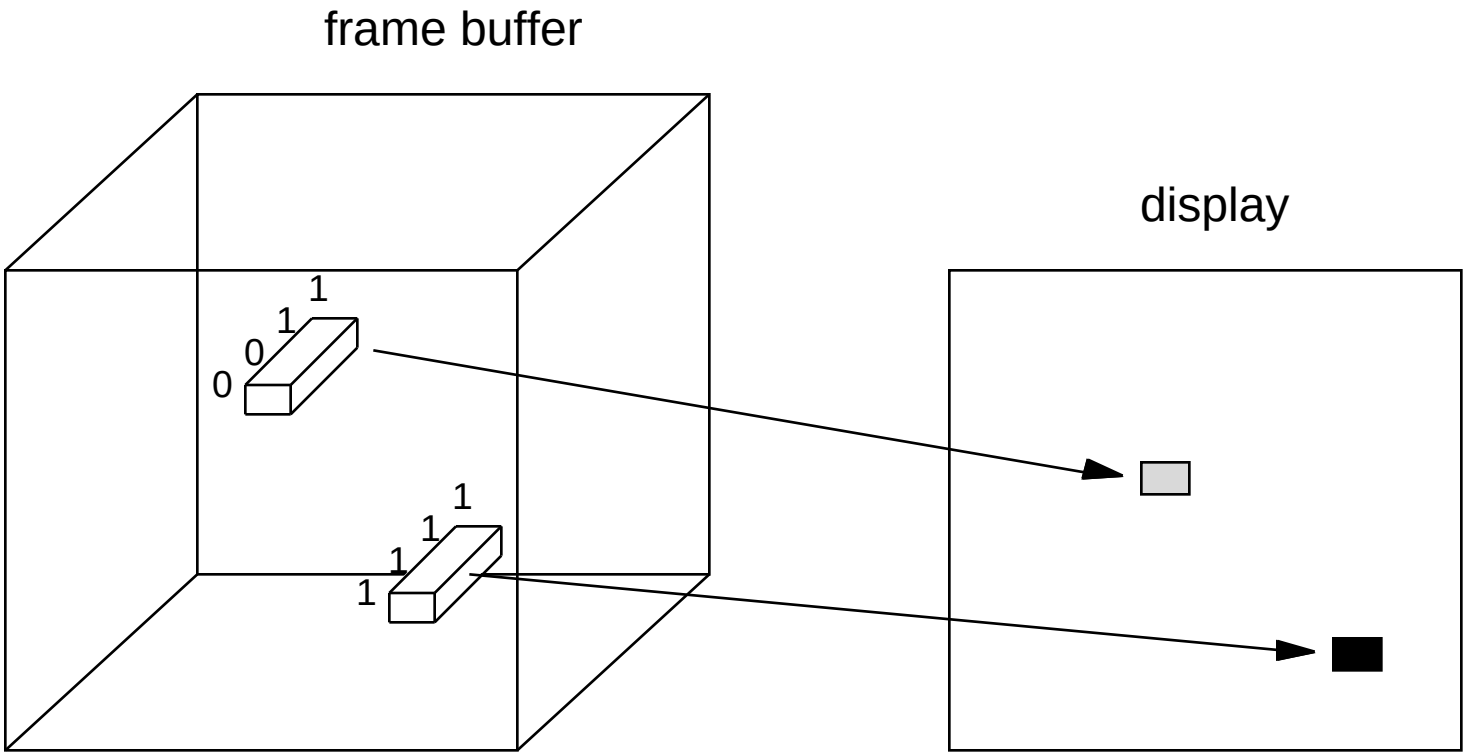
horizontal retrace

vertical retrace

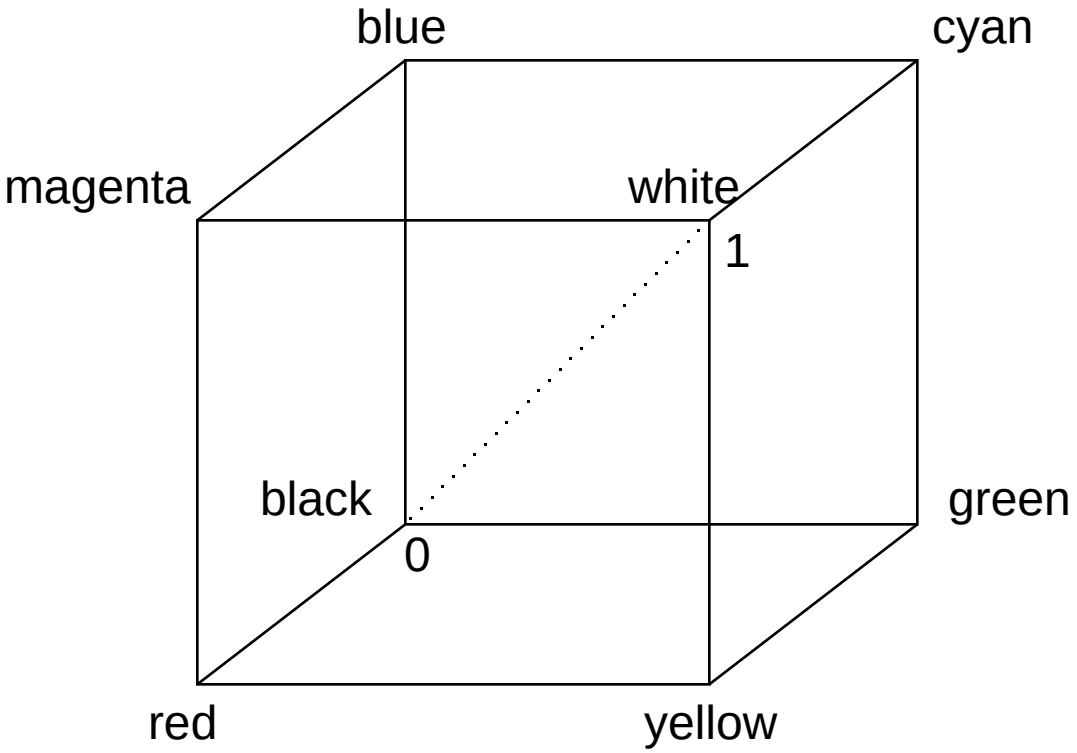
vertical:

horizontal:

Frame buffer (grayscale)



The RGB color space



Frame buffer with color map