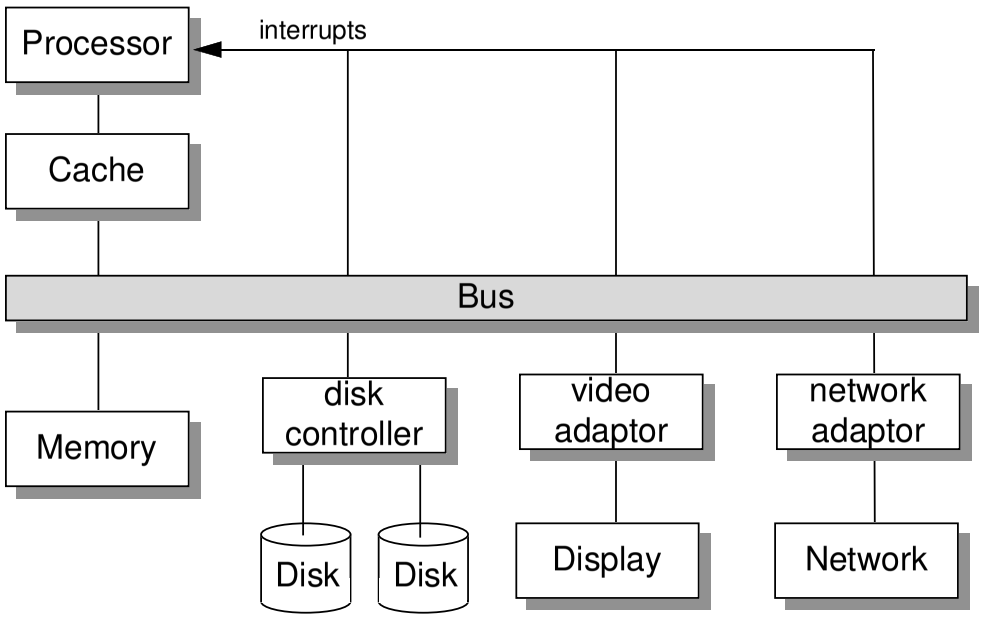
Buses

April 15, 1996

Topics:

- **types of buses**
- **bus protocols**
- **bus arbitration**
- **transferring data**

Computer system

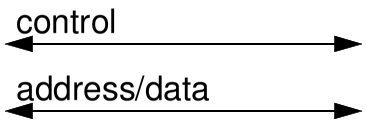


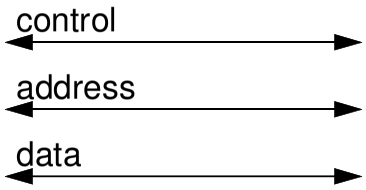
Anatomy of a bus

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

Consists of control and data/ address lines

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





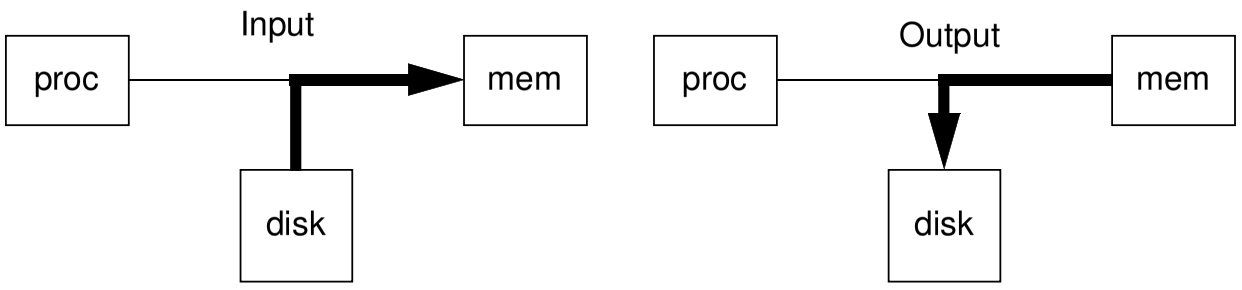
Bus transactions

Input transaction

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

Output transaction

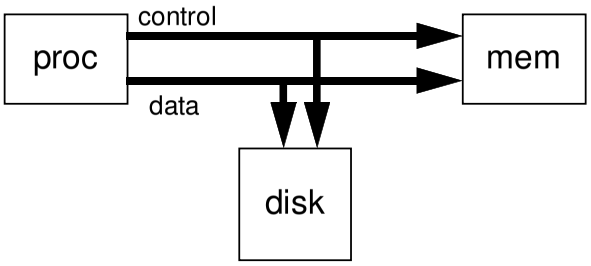
- **transfers data from memory or processor to device**



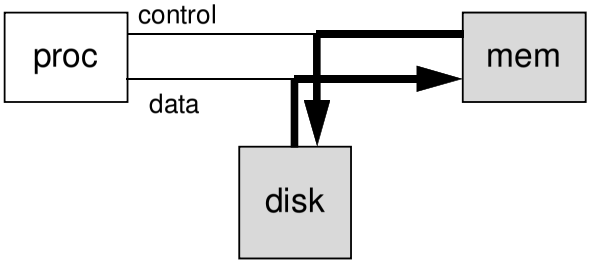
Input transaction

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

(assume disk address handled at a higher level)

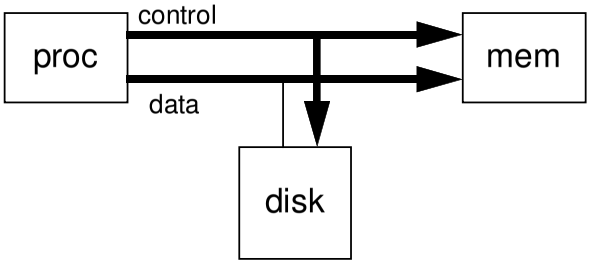


2. When memory is ready, it signals disk on control lines. Disk then transfers data.

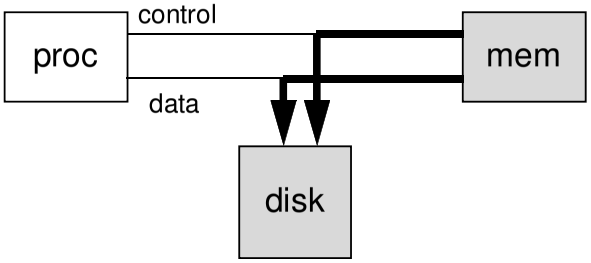


Output transaction

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 types

Processor-memory bus

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

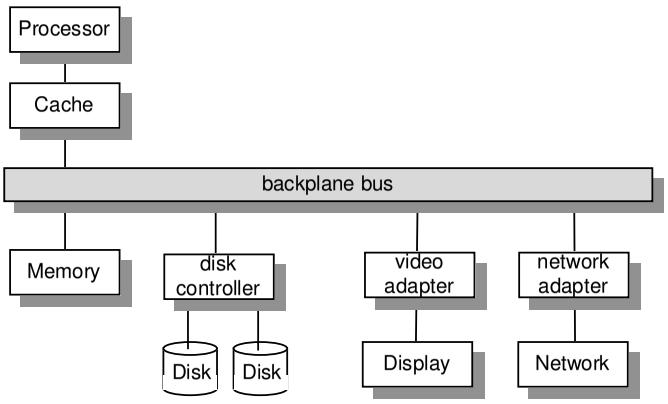
I/O bus

- **longer, slower, open**
- **unknown number of devices with different performance**
  - disk: 5 MB/s
  - 4x CD ROM: 160 KB/s
- **Examples: SCSI II, PCI, ISA, EISA**

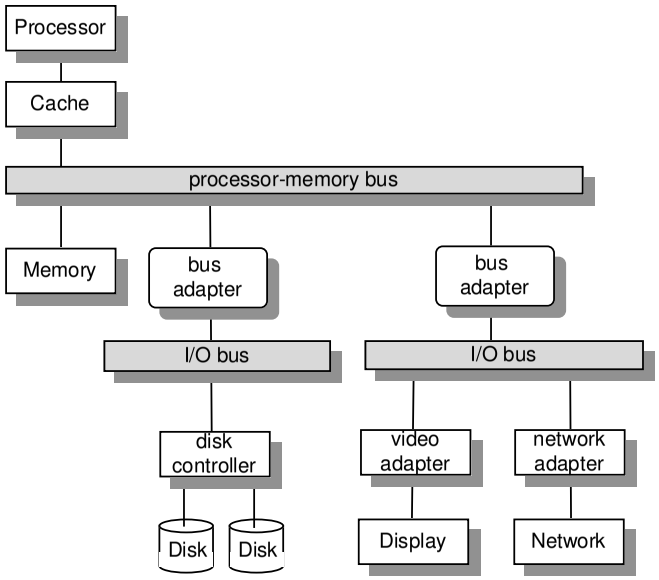
Backplane bus

- **processor, memory, and devices on same bus**
- **Examples: VME, NuBus**

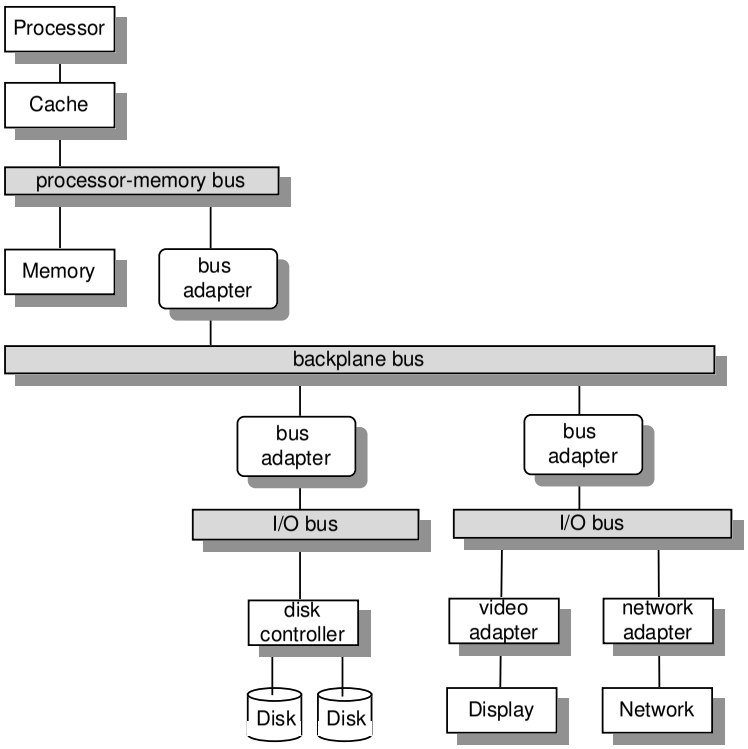
Bus organizations I



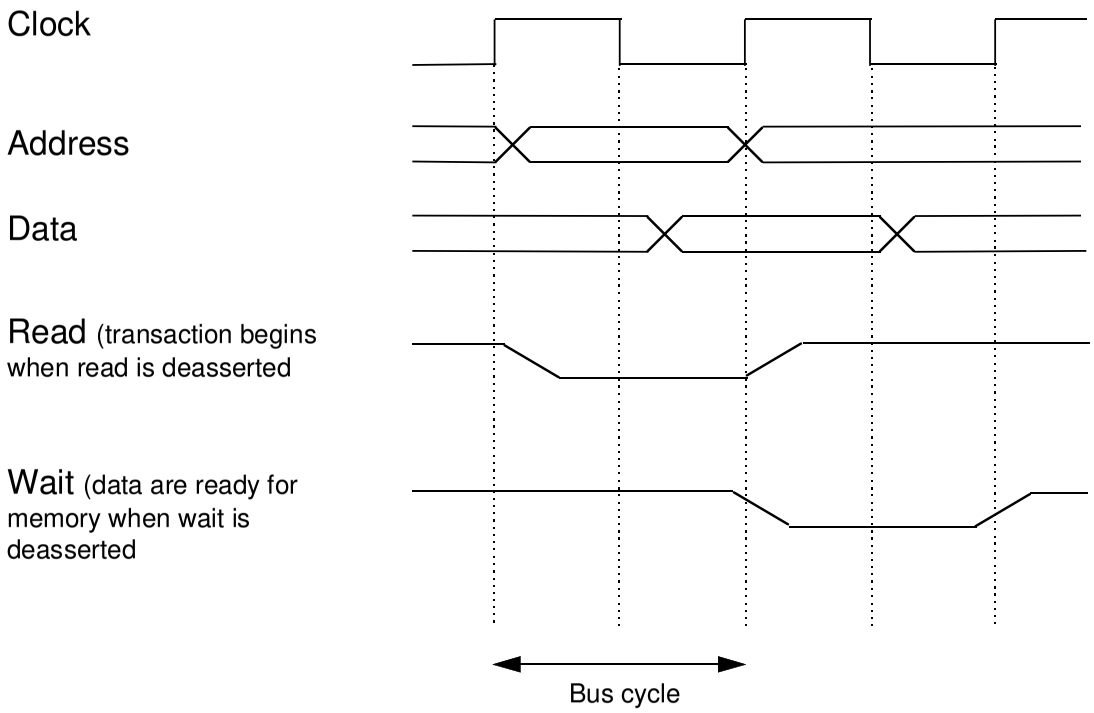
Bus organizations II



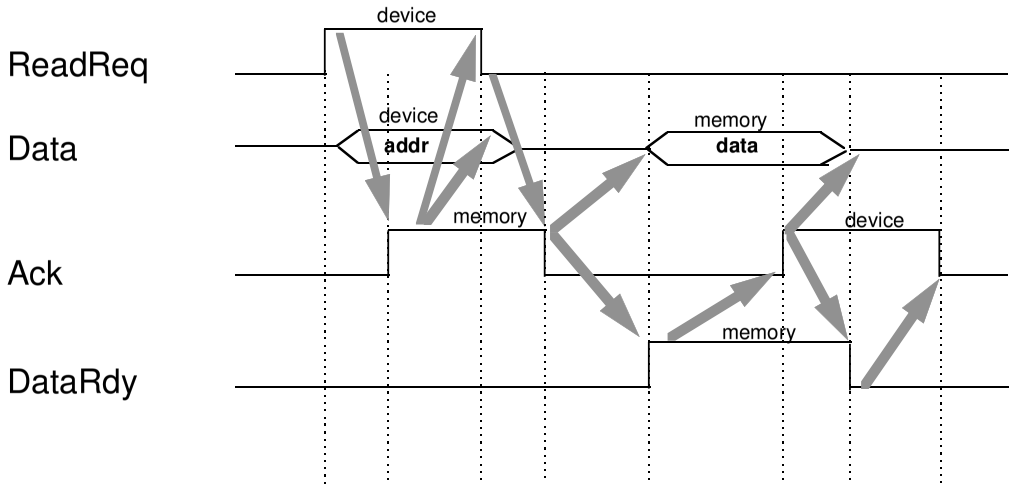
Bus organizations III



Synchronous input protocol



Asynchronous output protocol



Increasing bus bandwidth

Wider data path

- **disadvantage: more bus lines**

Separate vs multiplexed address and data lines

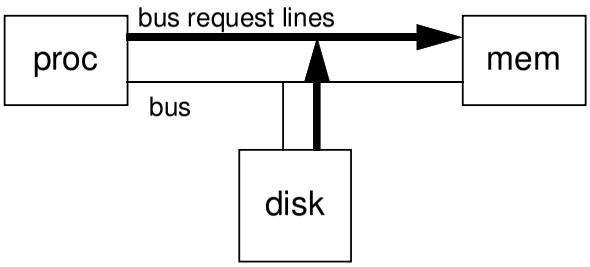
- **disadvantage: more bus lines**

bulk data transfers:

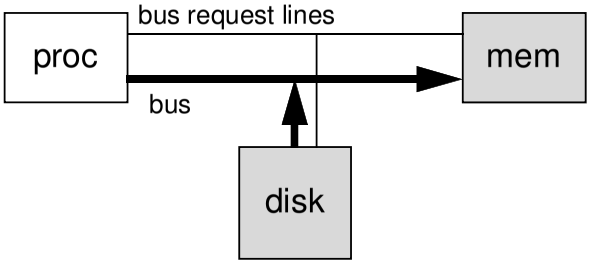
- **more than 1 data item per transaction**
- **<addr><len><data><data>...**
- **disadvantage: longer response time for other devices**

Bus arbitration (one master)

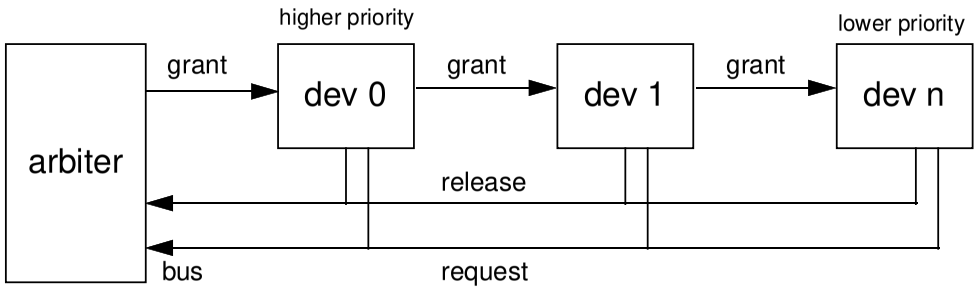
1. Device asks processor for permission to use the bus for an input transaction.



2. Processor initiates transaction on behalf of device.

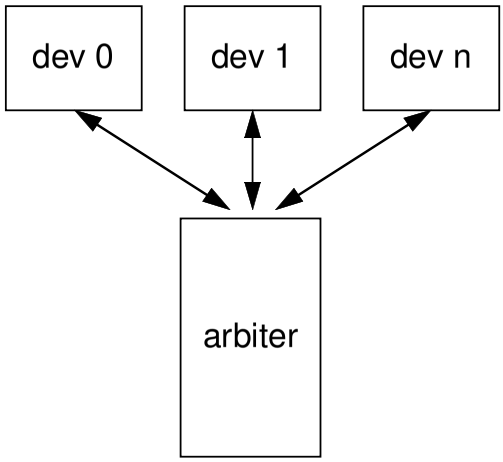


Daisy-chained arbitration



Not fair!

Centralized parallel arbitration



Not scalable!

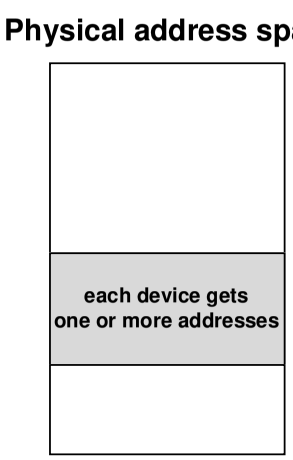
How processors send I/O commands

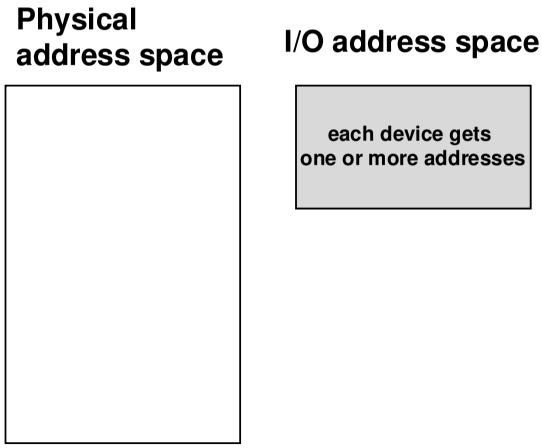
Memory mapped I/O

- 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

via an external interrupt

How data is transferred

Programmed I/O:

```
for (i=0; i<n; i++) {
    while (disk_stat_reg == NOTREADY) ;
    ld $2, disk_data_reg
}
```

DMA (direct memory access):

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