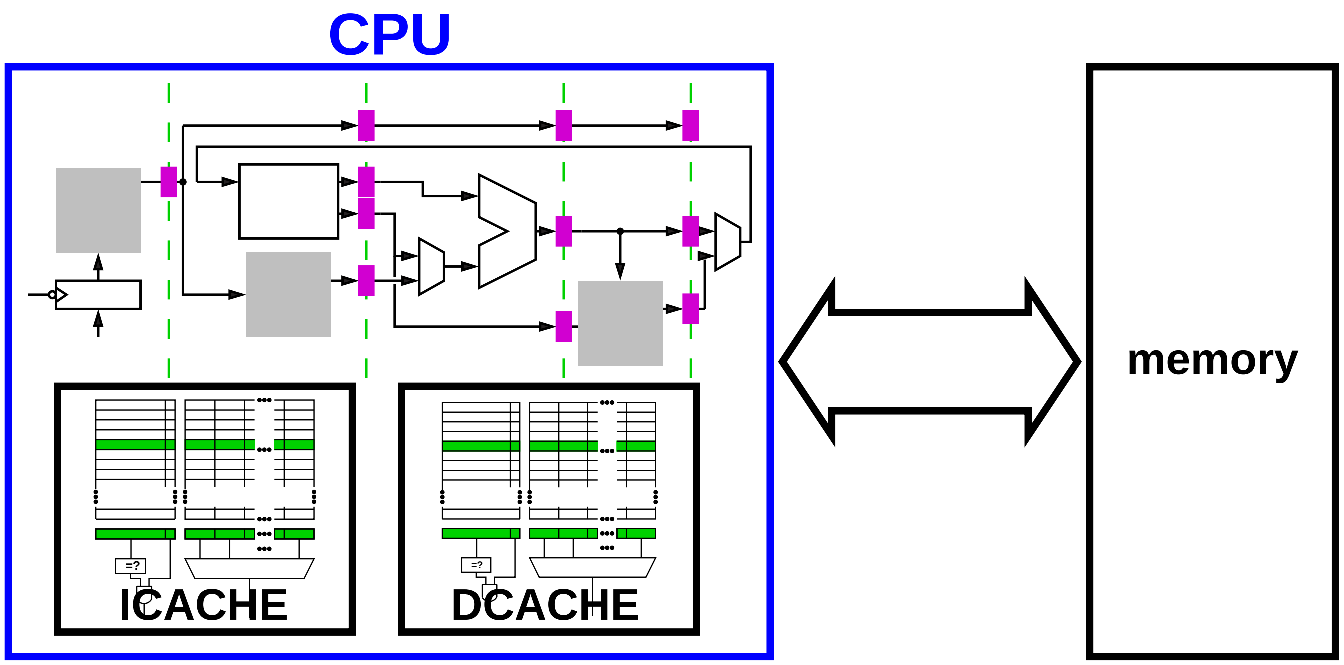
# Input/Output



How do we interface to other devices?

- keyboard, mouse, disk, network, graphics cards, floppy disk, printer, ...

## Operating System/Device Communication

Standard techniques for OS->device communication:

- Special instructions for I/O
  - CPU has additional output signals
  - Instruction specifies device
  - Instructions are privileged
- Memory-mapped I/O
  - Part of physical address space reserved for I/O
  - Read/writes interpreted as I/O commands
  - Protected by address translation

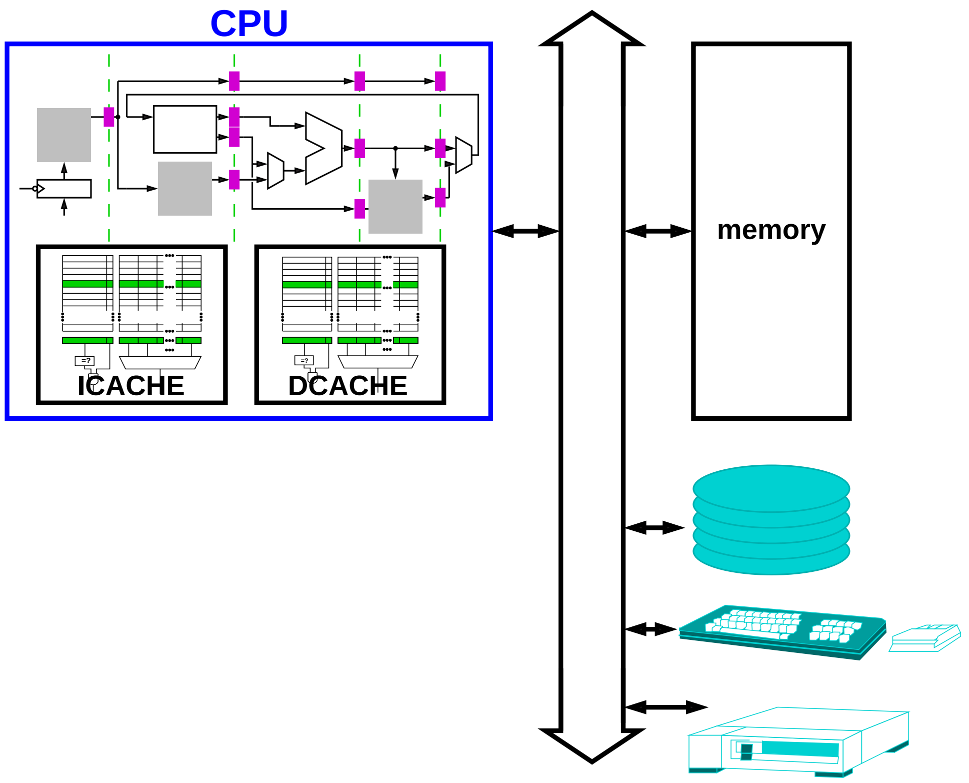
## Operating System/Device Communication

Standard techniques for device->OS communication:

- Polling
  - OS periodically checks I/O status
- Interrupts
  - Device sends interrupt
  - OS responds by communicating with device
  - Uses exception handling hardware

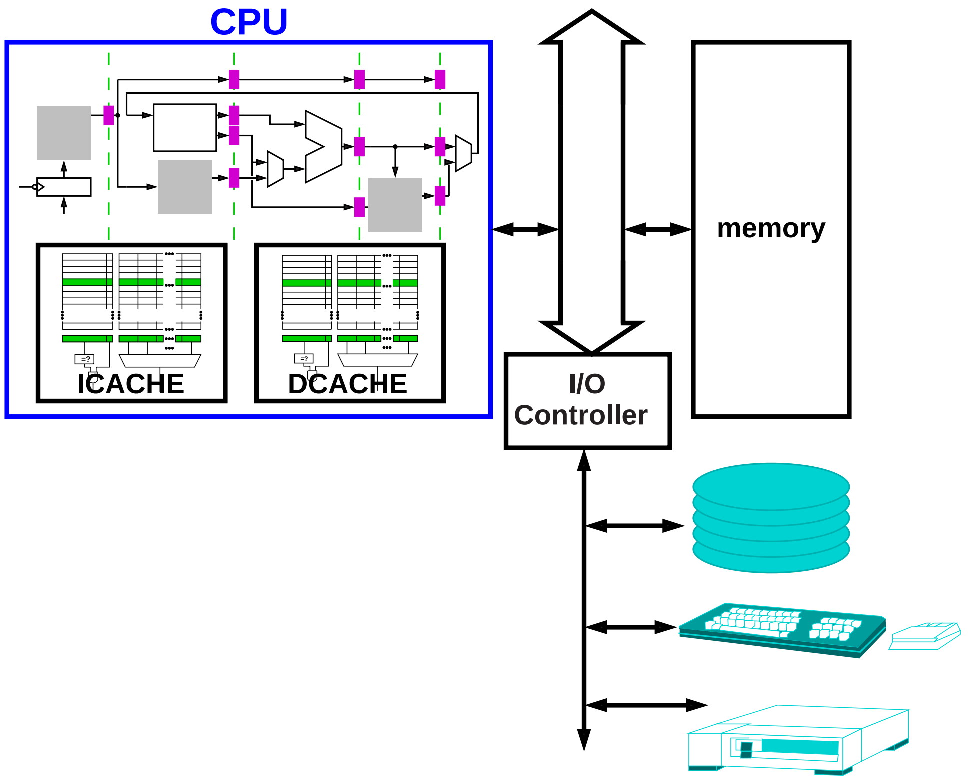
## I/O Controllers

We could do the following:



## I/O Controllers

Or:



## I/O Buses

What's a bus?

- A shared collection of wires with multiple senders/receivers
- Protocol, obeyed by senders and receivers
- Simple, broadcast mechanism, serializes accesses
- Can be slow, serializes accesses

Standards to ensure compatibility of devices

PCI, USB, SCSI, ISA, EISA, VESA, NuBUS

*The nice thing about standards is that there are so many to choose from – Andrew Tanenbaum*

## Bus Parameters

Standard parameters:

- Bus width: number of wires used, separate control/data?
- Data width: number of bits per transfer
- Transfer size: number of words per bus transaction
- Bus *masters*: single, multiple
- Split transaction
- Synchronous/asynchronous

## Buses

General protocol sequence:

- Arbitrate for bus mastership
- Winner sends address to slave
- Data transferred between master and slave

Different schemes for arbitration (usually with priorities)

Standard bus in desktops:

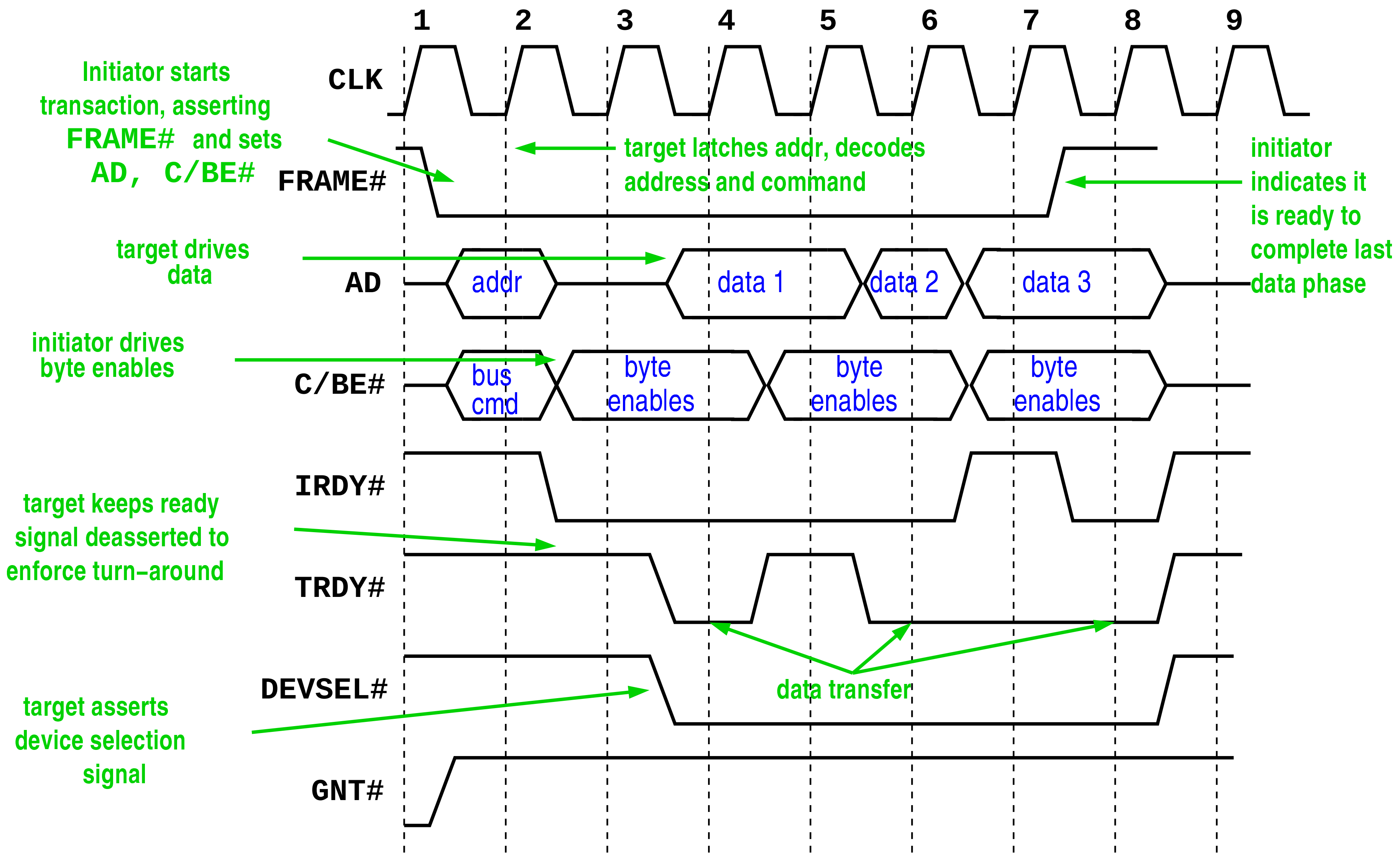
- PCI (peripheral component interconnect)

## PCI Characteristics

- 33/66 MHz clock (`CLK`)
- Central arbitration (`REQ#, GNT#`)
  overlapped with previous transaction
- Multiplexed address/data (32/64 lines) (`AD`)
- General protocol
  - bus command (`C/BE#`)
  - addr handshake and duration (`FRAME#, TRDY#`)
  - data width (byte enable) (`C/BE#`)
  - variable-length data handshake (`IRDY#, TRDY#`)

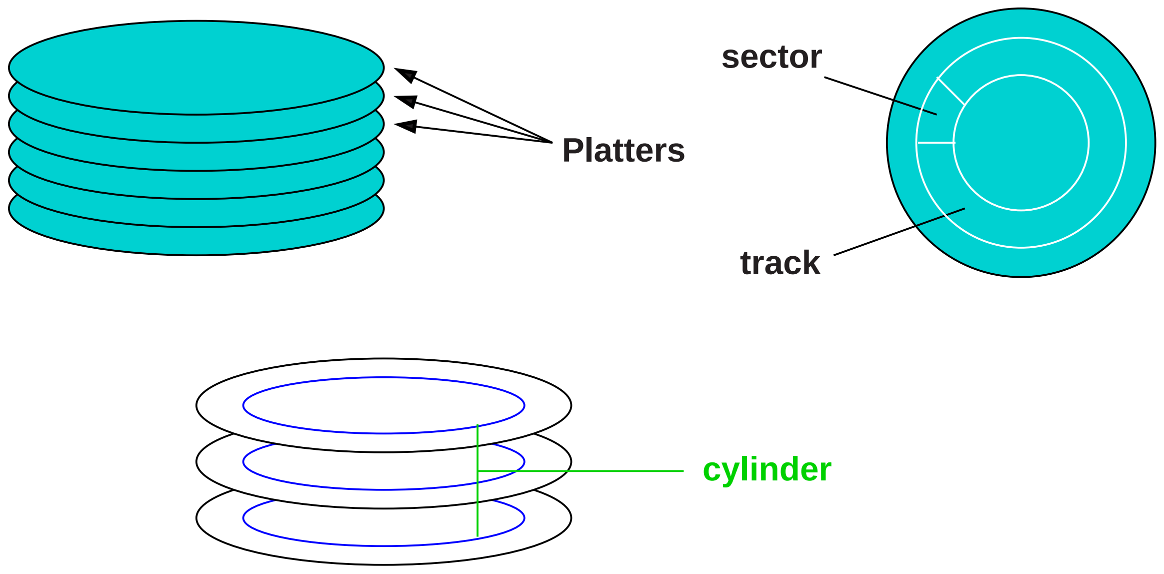
Max BW: 132MB/s base

# PCI Read



1
2
3
4
5
6
7
8
9
CLK
FRAME#
AD
C/BE#
IRDY#
TRDY#
DEVSEL#
GNT#
Initiator starts transaction, asserting FRAME# and sets AD, C/BE#
target latches addr, decodes address and command
initiator indicates it is ready to complete last data phase
target drives data
addr
data 1
data 2
data 3
initiator drives byte enables
bus cmd
byte enables
byte enables
byte enables
target keeps ready signal deasserted to enforce turn-around
data transfer
target asserts device selection signal

# Disk Organization



Sizes are growing (a 9GB disk is $10^9$ bytes)

- IBM: 35.3 billion bits/sq. inch (Oct 1999)
- Production: >10 billion bits/sq. inch

## Disk Performance

Disks rotate at 7200-15000rpm (6-8ms per revolution)

Rotational latency:

- Average latency to a sector: 3-4ms

Transfer time:

- Transfer size is about a sector (512B to 1KB)
- Typical values: 40-160MB/s

Read time = seek time + rotation latency + transfer time + controller overhead

## I/O Transfers

Normal mode:

- Operating system requests transfer from disk
- Disk wants to write to memory
- As described, needs OS intervention to write each word to memory!
- Observation: I/O controller is on the memory bus...

What if we allow the device to write directly to memory?

(Notify the CPU when the transfer is complete)

## DMA

DMA: direct memory access

- OS asks device to read data into memory
- Device transfers data directly to memory
  - Might interfere with cache miss requests!
  - What if the address is cached?
  - What happens if the address is not mapped?!
    - Can use kernel addresses (requires copy)
    - Notify OS: pages are *pinned*
    - TLB shootdown?
- Device sends interrupt when done