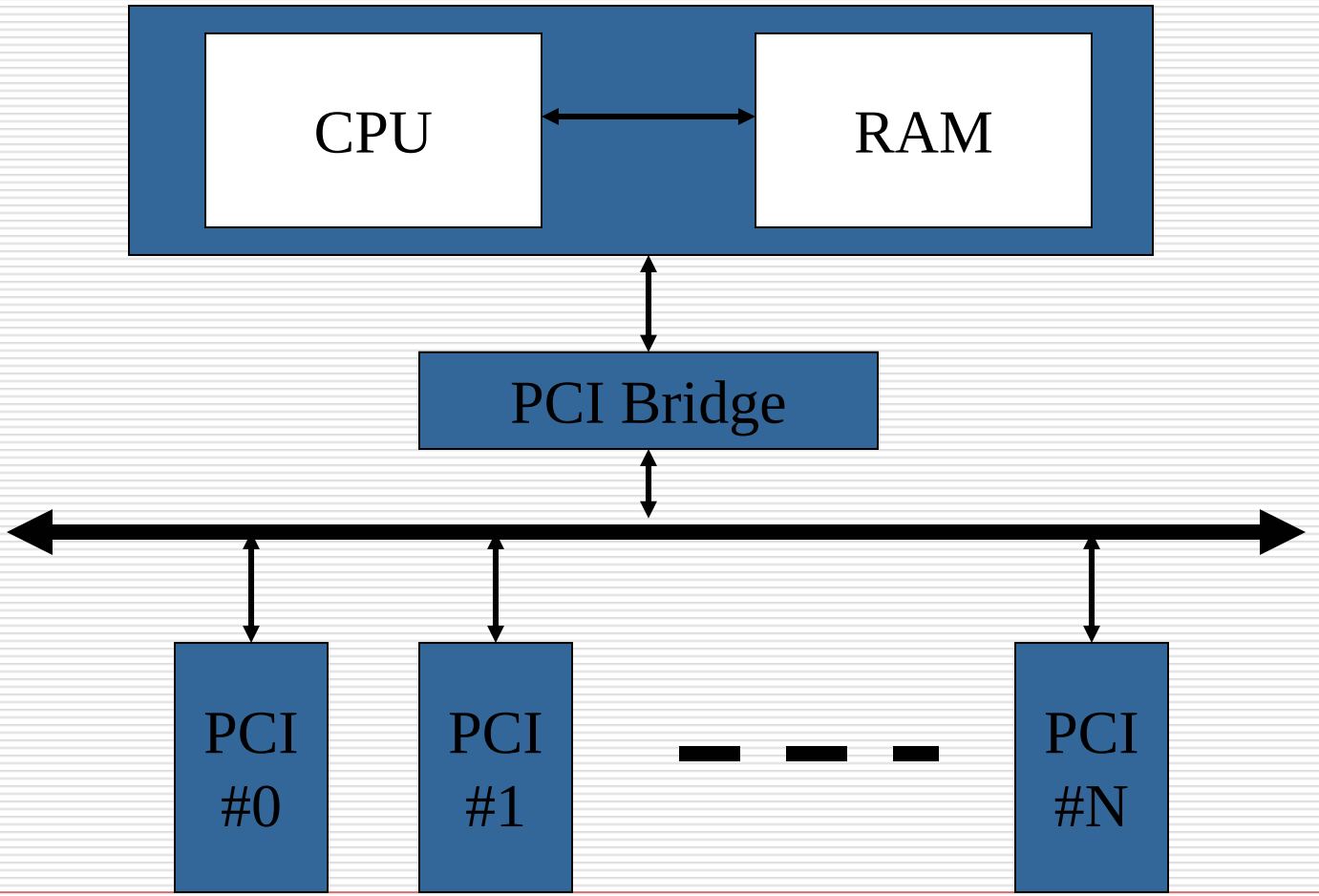
CS 4405

PCI Devices

PCI Devices

- PCI stands for *Peripheral Component Interconnect*.
- One nice feature of the PCI bus is complete separation of the bus from the CPU.
  - This allows the bus to be used under different hardware platforms without requiring change of the processor design.
- To make up for this lack of control, there exists the PCI bridge.

PCI Schematic



Breeds of PCI

- Due to the independence of the bus from the CPU, PCI has more flexibility to perform at different rates.
  - Speed: 33 Mhz, 66 Mhz, 133 Mhz, ...
  - Data width: 32-bit, 64-bit, 128 bit, ...
- The data width is a physical determination with a PCI card.
- The speed is configurable based on what the PCI card and PCI bus can perform.
  - Average PC's perform under 32-bit/33Mhz.

Data Transfer

- With the PCI bus, the bus lines are shared between data and addresses.
- To accomplish this, data and addresses are sent alternating over the bus.
- Another solution is the PCI bridge can divide the bus between an address and data.
- Another solution is PCI burst.
  - The address is sent, followed by a data block.
  - Addresses are automagically incremented by bridge and adapter.

PCI Bios

- A configuration address space of 256 bytes is provided for each device.
- The PCI Bios fills in this address space for each device.
  - The first 64 bytes are specified by the PCI specification.
  - The remaining 192 bytes are for the manufacturer.
- The PCI Bios can also give the PCI device an IRQ.

Identifying PCI resources

- Every device on the PCI bus has a definite address.
  - The PCI bus itself has an identifier (there could be several PCI busses).
  - The slot the PCI card is plugged into has an identifier.
  - The device itself has function numbers for each subunit (e.g. PCI SCSI controller).
- Moving the card changes address.
  - This is why windows discovers new hardware!

Finding Your Device

- PCI devices are interesting because, the driver must find the device.
- This is a result of different factors including which slot you plug the card into and what interrupt the PCI Bios assigned the device.
- Initializing the device driver requires finding the device.

Finding Your Device (cont.)

- Typically a kernel maintains a list structure of ALL PCI devices that are physically connected to the machine
  - `pci_find_device()` - find a specific device in the PCI address space. Can be called multiple times with offset to multiple devices.
  - `pci_find_subsys()` - allows the specification of a subsystem vendor and device code.
  - `pci_find_class()` - looks for a device class (e.g. any network card).

Operating System Support

- The kernel should provide several macros and functions to handle the PCI configuration space.
  - The first 64 bytes
- These macros can be used to gather important information that the device programmer may need.
  - `PCI_VENDOR_ID`
  - `PCI_DEVICE_ID`
  - ...

Configuration Space

- The configuration space contains data that is specific to the device.
- The kernel needs to support accessing this information on the device:
  - `pci_read_config_word()` - for reading.
  - `pci_write_config_word()` - for writing.
- Only allowed to play with the vendor specific area.

PCI Device Memory

- Most communication between a PCI device and the CPU is through memory that exists on the PCI card.
- Before the memory can be used by the device driver it must first be allocated.
  - `request_mem_region()`
- Once allocated, the driver can perform an `ioremap()` and `iounmap()` to map the memory into the virtual address space of CPU.

Where's the Byte?

- PCI device memory adds a new addressing space.
  - Physical address, virtual address, and now the bus address (what address does the PCI card memory reside at?).
- For the x86, this is just an extension of the physical address.
- For other architectures this is not the case.

PCI Addressing

- The kernel needs to provide macros for the driver to use to ensure proper conversion of addressing between addressing spaces.
  - `virtual_to_physical()`
  - `physical_to_virtual()`
- This results in the memory from the device appearing no different than any other memory in the system.