PENNSTATE

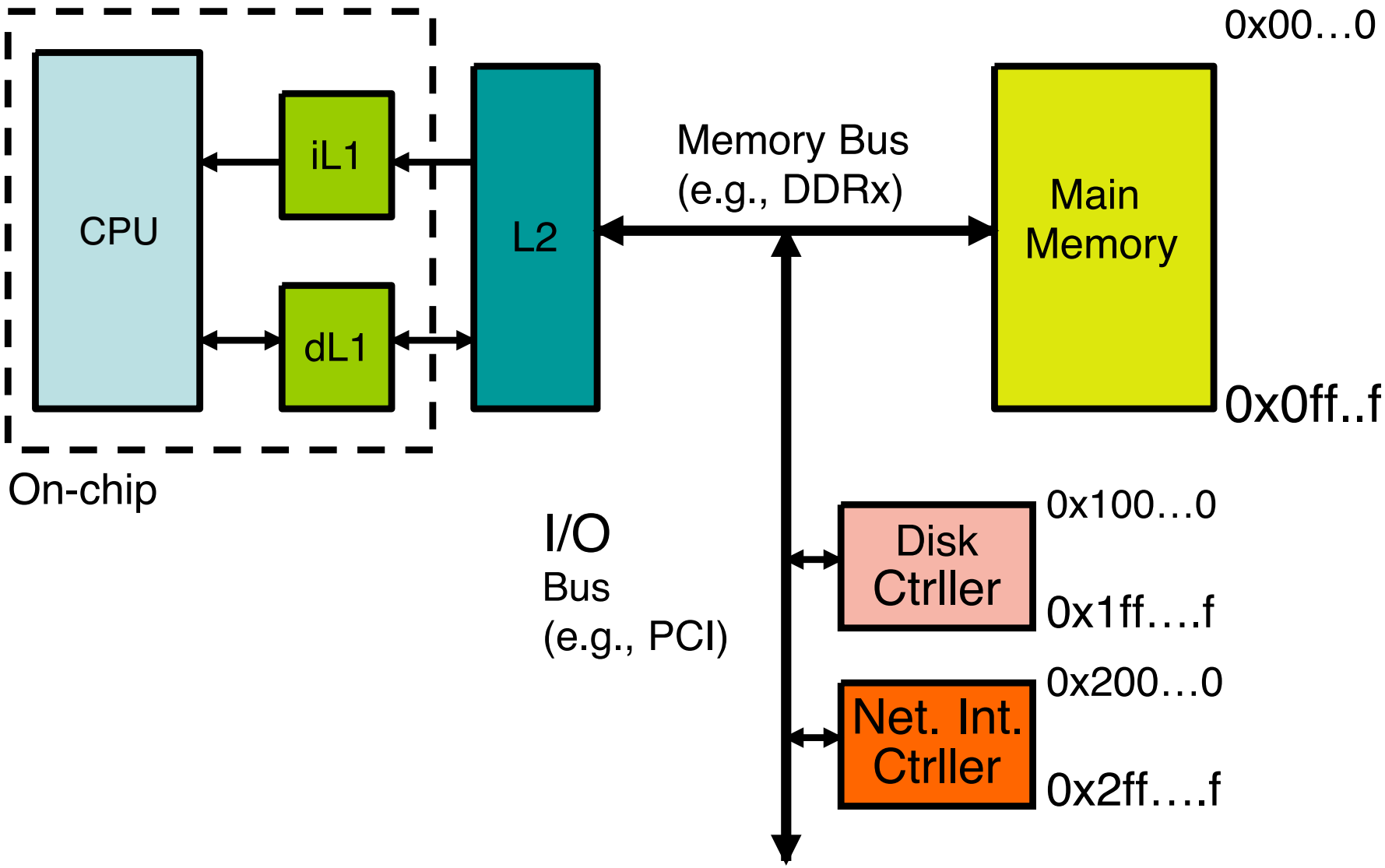
Input-Output

PENNSTATE

OS role in I/O

- **Share the same device across different processes/users**
- **User does not see the details of how hardware works**
- **Device-independent interface to provide uniformity across devices.**

I/O Peripherals



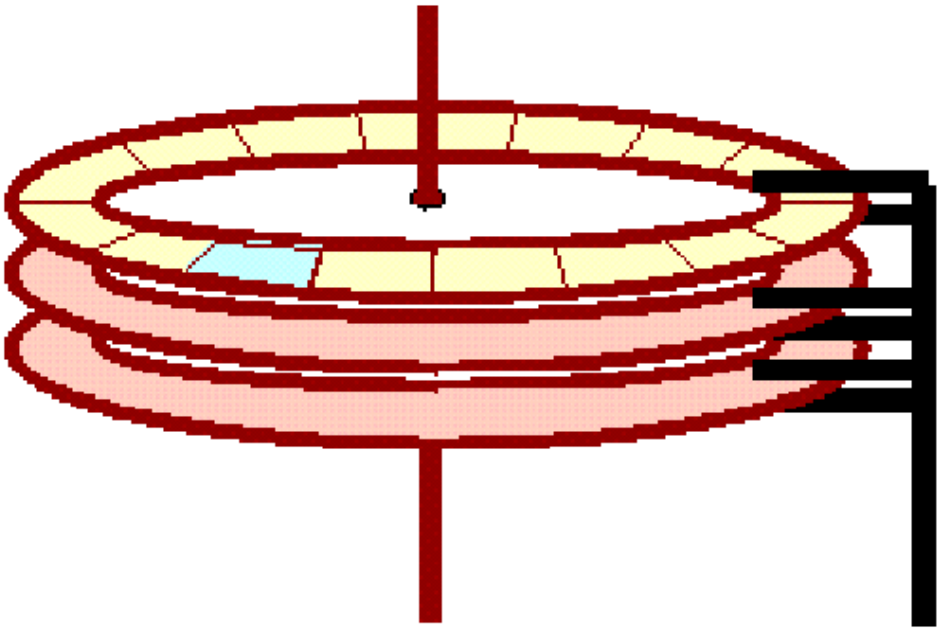
PENNSTATE

Disks: a quick tutorial

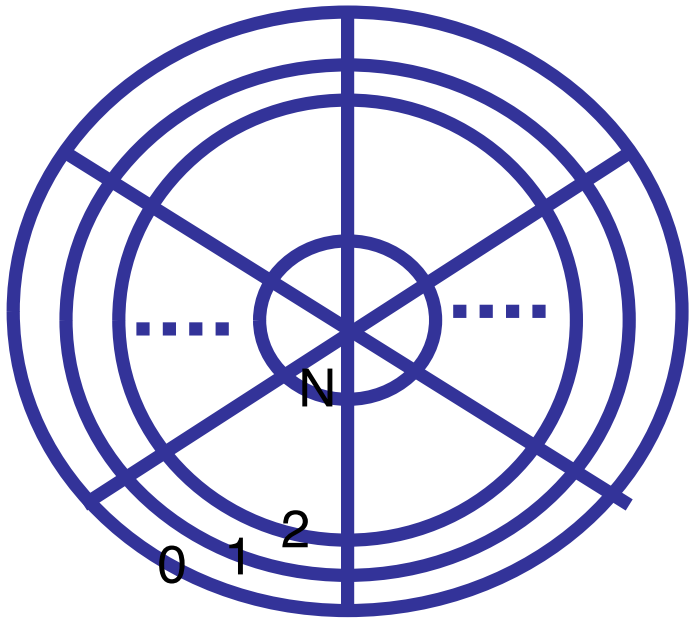
- A disk has platters, each with 2 surfaces, mounted on a spindle assembly.
- Each surface has circular tracks.
- Each track has sectors.
- The same track across all the surfaces of all platters, together constitute a cylinder.
- There is a head for each surface of each platter.
- All the heads are again mounted on a single assembly, and thus need to move in unison, i.e., all heads will have to be on the same cylinder at any time.

A pictorial view

Platters and heads



Tracks and sectors on a surface



PENNSTATE

Disk Access Costs

- **Actions to be performed:**
  - **(a) seek to a specific cylinder**
  - **(b) wait for the given sector(s) to come under the head**
  - **(c) transfer the data**
- **Of these (a) is typically the most dominant (running into milliseconds)**
- **(a) is proportional to the seek distance in cylinders, which is one of the optimization goals of disk scheduling**

PENNSTATE

Disk Scheduling

- **Waiting requests for cylinders:**
  - **1, 36, 16, 34, 9, 12**
- **Current head position is cylinder 11**

- **First Come First Serve (FCFS)**
  - **Service Order: 1,36,16,34,9,12**
  - **Seek cost = 10+35+20+18+25+3=111 cylinders**
- **Shortest Seek Time First (SSTF)**
  - **Service Order: 12,9,16,1,34,36**
  - **Seek cost = 1+3+7+15+33+2=61 cylinders**

PENNSTATE

- **What is the problem with SSTF?**
  - **May not be very fair (and can lead to starvation)**

- **Analogy between this problem and an elevator serving requests of clients waiting on different floors of a building.**

PENNSTATE

SCAN Algorithm

- The disk head moves in one direction servicing requests until it gets to the end of the disk, where the head movement is reversed and servicing continues.
- Consequently, there is a bound (twice the number of cylinders) on the number of cylinders a head traverses before getting to a request.
- Waiting requests for cylinders:
  - 1, 36, 16, 34, 9, 12
- Current head position is cylinder 11 and is going in increasing direction.
- Number of cylinders on disk = 100 (0 … 99)
- Service order: 12, 16, 34, 36, 9, 1
- Seek cost: 1+4+18+2+63+90+8= 186 cylinders

PENNSTATE

- **Some "unfairness" in SCAN**
  - **Requests in the center cylinders would have a shorter wait time.**

PENNSTATE

Circular-SCAN (C-SCAN)

- **To provide a more uniform wait time than SCAN.**
- **Similar to SCAN, except that after reversing direction, it goes back all the way to track 0 without servicing any requests and starts again. i.e., it serves request only in the "up" direction. – resembles a circular list**

PENNSTATE

C-SCAN

- **Waiting requests for cylinders:**
  - **1, 36, 16, 34, 9, 12**
- **Current head position is cylinder 11, and is going in increasing direction.**
- **Number of cylinders on disk = 100 (0 … 99)**
- **Service order: 12, 16, 34, 36, 1, 9**
- **Seek cost: 1+4+18+2+63+99+1+8= 197 cylinders**

LOOK

- Variant of SCAN where head only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.
- Waiting requests for cylinders:
  - 1, 36, 16, 34, 9, 12
- Current head position is cylinder 11, and is going in increasing direction.
- Number of cylinders on disk = 100 (0 … 99)
- Service order: 12, 16, 34, 36, 9, 1
- Seek cost: 1+4+18+2+27+8= 60 cylinders
- Note that it is not necessary to be lower than SSTF.

PENNSTATE

C-LOOK

- Variant of C-SCAN where head only goes as far as the last request in each direction, then reverses direction immediately, without first going all the way to the end of the disk.
- Waiting requests for cylinders:
  - 1, 36, 16, 34, 9, 12
- Current head position is cylinder 11, and is going in increasing direction.
- Number of cylinders on disk = 100 (0 … 99)
- Service order: 12, 16, 34, 36, 1, 9
- Seek cost: 1+4+18+2+35+8= 68 cylinders

PENNSTATE

I/O Software

- **A layered approach:**
  - **Lowest layer (device dependent): Device drivers**
  - **Middle layer: Device independent OS software**
  - **High level: User-level software/libraries**
- **The first 2 are part of the kernel.**

PENNSTATE

Device-independent OS Layer

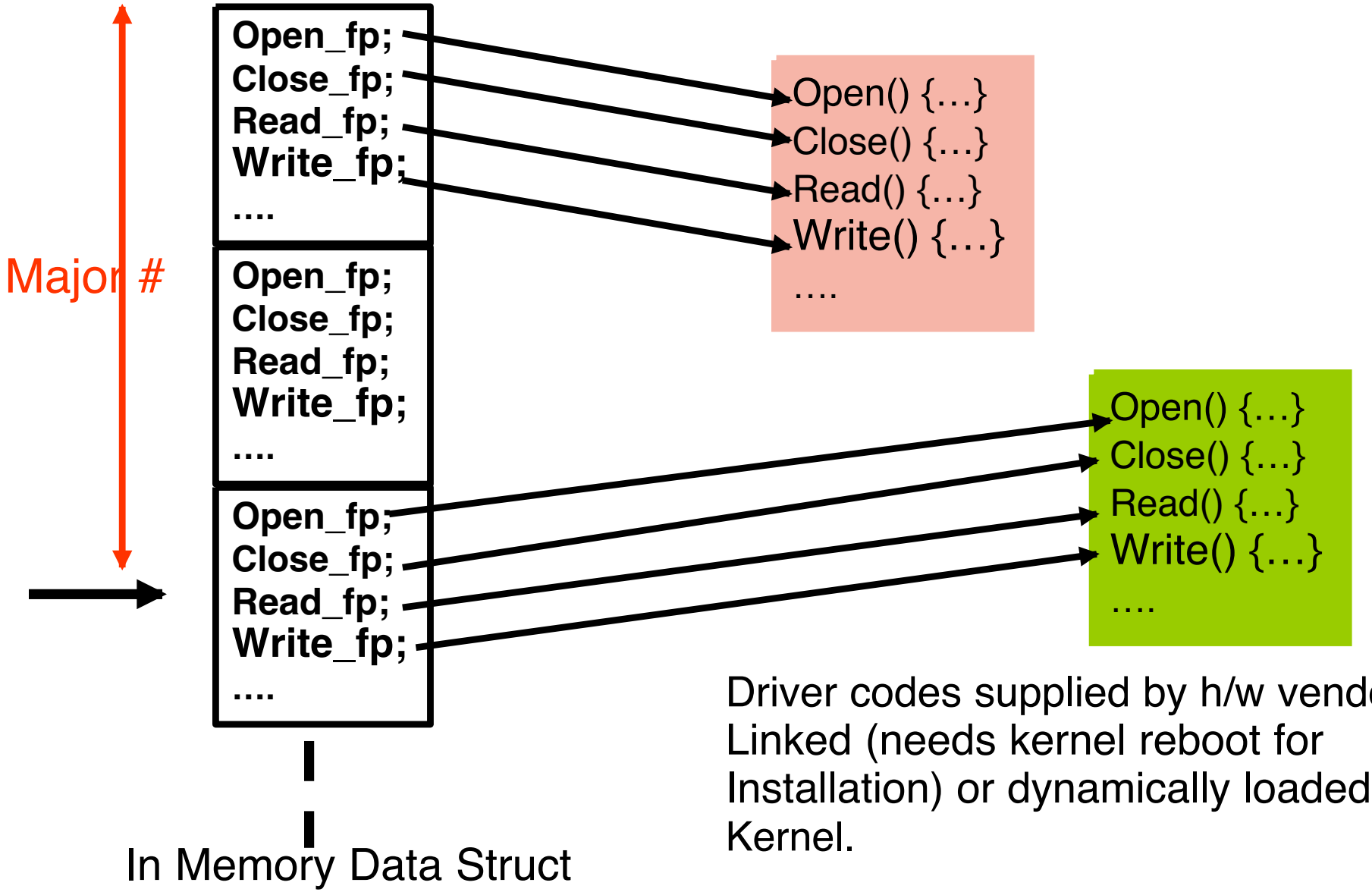
- **Device naming and protection**
  - **Each device is given a (major #,minor #) – present in the i-node for that device**
  - **Major # identifies the driver**
  - **Minor # is passed on to the driver (to handle sub-devices)**
- **Does buffering/caching**
- **Uses a device-independent block size**
- **Handles error reporting in a device-independent fashion.**

PENNSTATE

Putting things together (UNIX)

- User calls open("/dev/tty0","w") which is a system call.
- OS traverses file system to find the i-node of tty0.
- This should contain the (major #, minor #).
- Check permissions to make sure it is allowed.
- An entry is created in OFDT, and a handle is returned to the user.
- When user calls write(fd, ….) subsequently, index into OFDT, get major/minor #s.

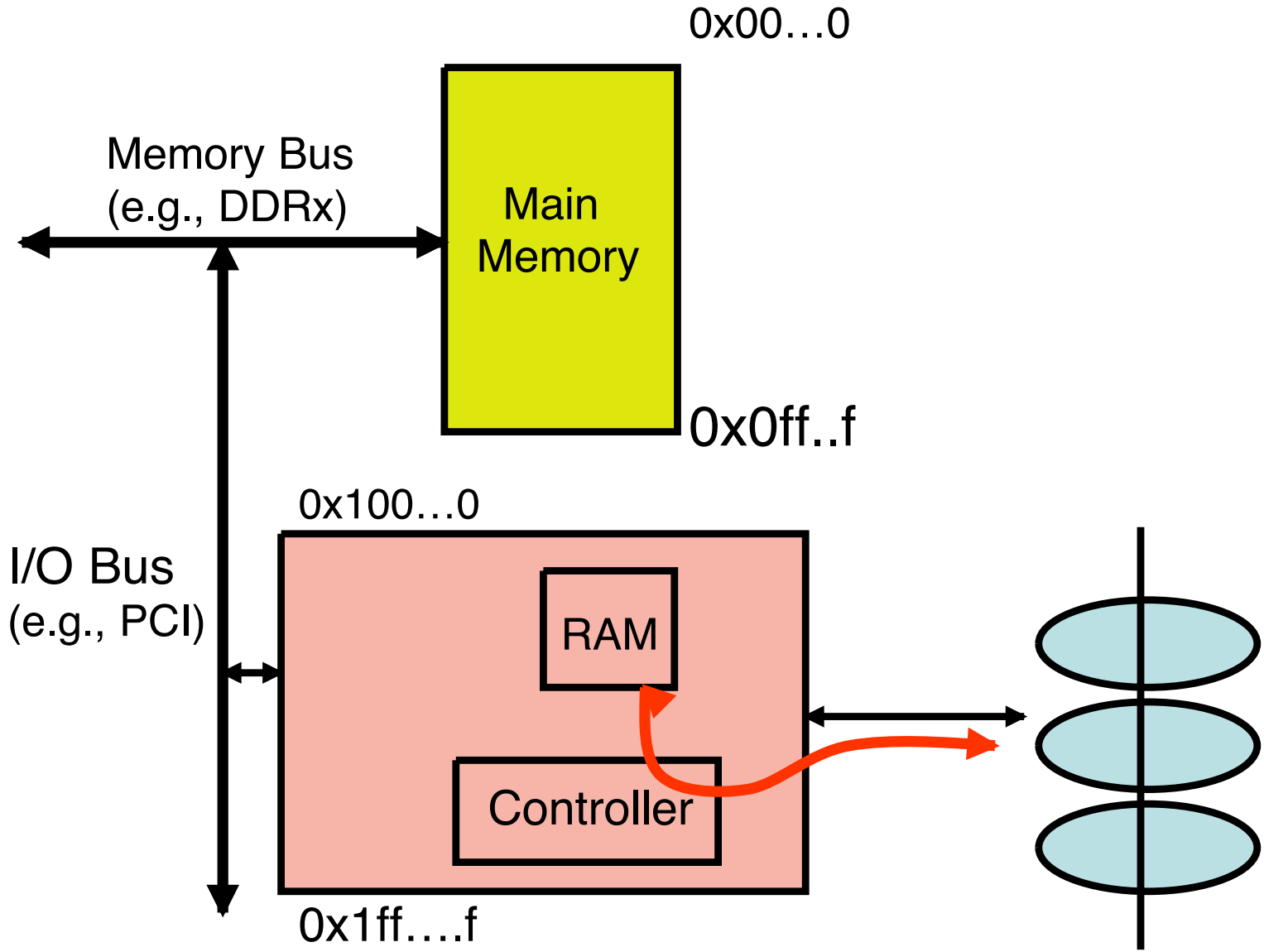
Getting to the driver routine



Driver codes supplied by h/w vendors Linked (needs kernel reboot for Installation) or dynamically loaded into Kernel.

- Copy the bytes pointed to by the pointer given by user, into a kernel “pinned” (which is not going to be paged out) buffer.
- Use the above data structure, to find the relevant driver’s write() routine, and call it with the pinned buffer address, and other relevant parameters.
- For a write, one can possibly return back to user even if the write has not propagated. On a read (for an input device), the driver would program the device, and block the activity till the interrupt.

Consider a disk device …



Reading a sector from disk

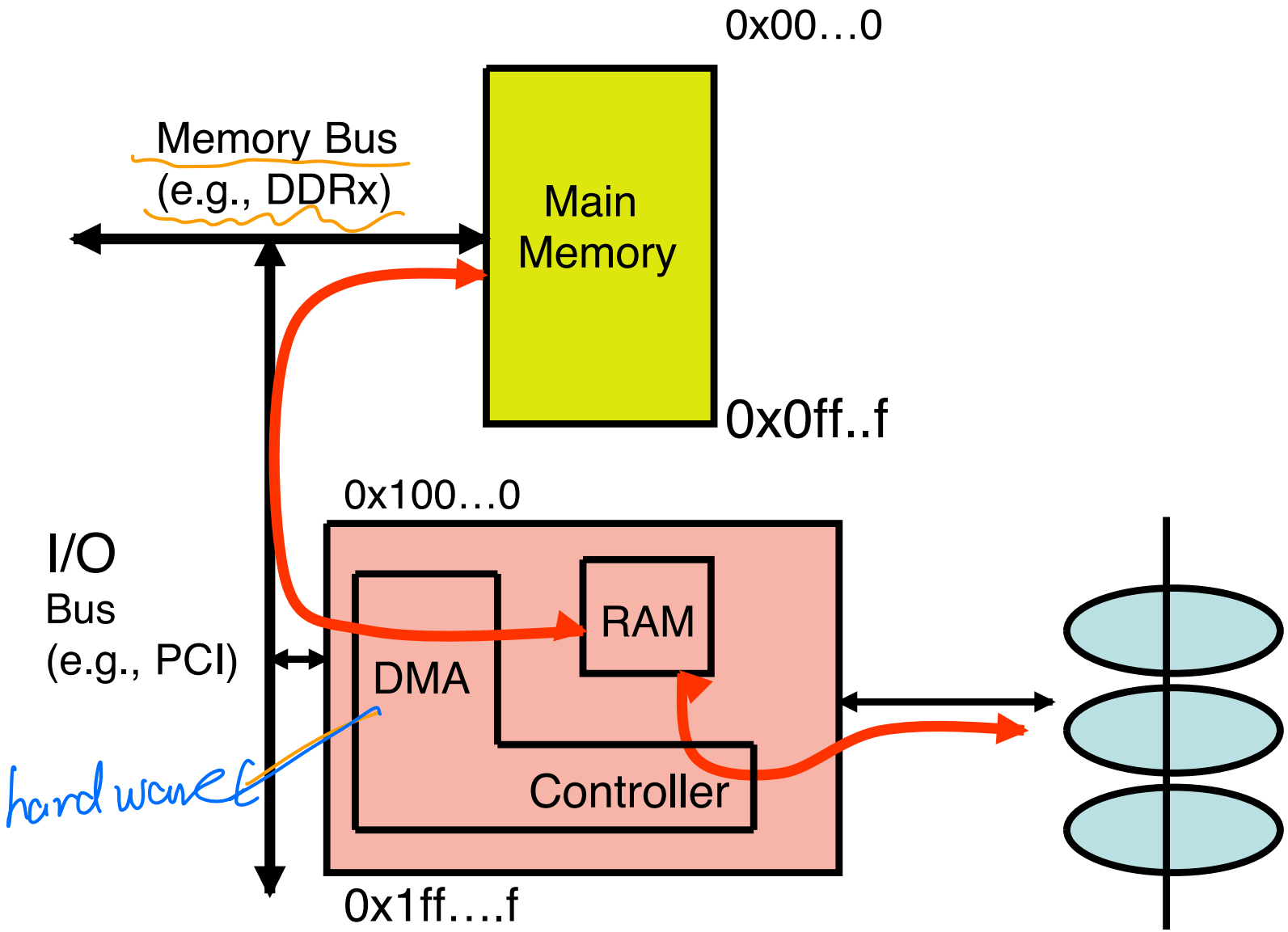
```
Store [Command_Reg], READ_COMMAND
Store [Track_Reg], Track #
Store [Sector_Reg], Sector #

/* Device starts operation */


L:   Load R, [Status_Reg]
     cmp R, 0
      jeq

/* Data now on memory of card */

For i = 1 to sectorsize
    Memtarget[i] = MemOnCard[i]
```

You don't want to do this! Instead, block/switch to other process and let an interrupt wake you up.

This is again a lot of overhead to ask the main CPU to do!

DMA ← draw data quickly into mem.
will doing this to improve

DMA engine to offload work of copying



DIRECT :

MEMORY :

ACCESS :

```
Store [Command_Reg], READ_COMMAND
Store [Track_Reg], Track #
Store [Sector_Reg], Sector #
Store [Memory_Address_Reg], Address

/* Device starts operation */

P(disk_request);

/* Operation complete and data is
    now in required memory locations*/
```

Assuming an integrated DMA and disk ctrller.

Called when DMA raises interrupt after Completion of transfer

```
ISR() {
    V(disk_request);
}
```

PENNSTATE

Issues to consider

- **What is purpose of RAM on card?**
  - **To address the speed mismatch between the bit stream coming from disk and the transfer to main memory.**

Issues to consider (contd.)

- **When we program the DMA engine with address of transfer (Store [Memory_Address_Reg], Address), is Address virtual or physical?**
  - **It has to be a physical address, since the addresses generated by the DMA do NOT go through the MMU (address translation).**
  - **But since it is the OS programming the DMA, the physical address is available and NOT a problem.**
  - **You do NOT want to give this option to user programs.**
  - **Also, the address needs to be "pinned" (cannot be evicted) in memory.**