OSTEP Chapter 36

ECE 3600, Fall 2022

Table of Contents

1. I/O Hardware and Software
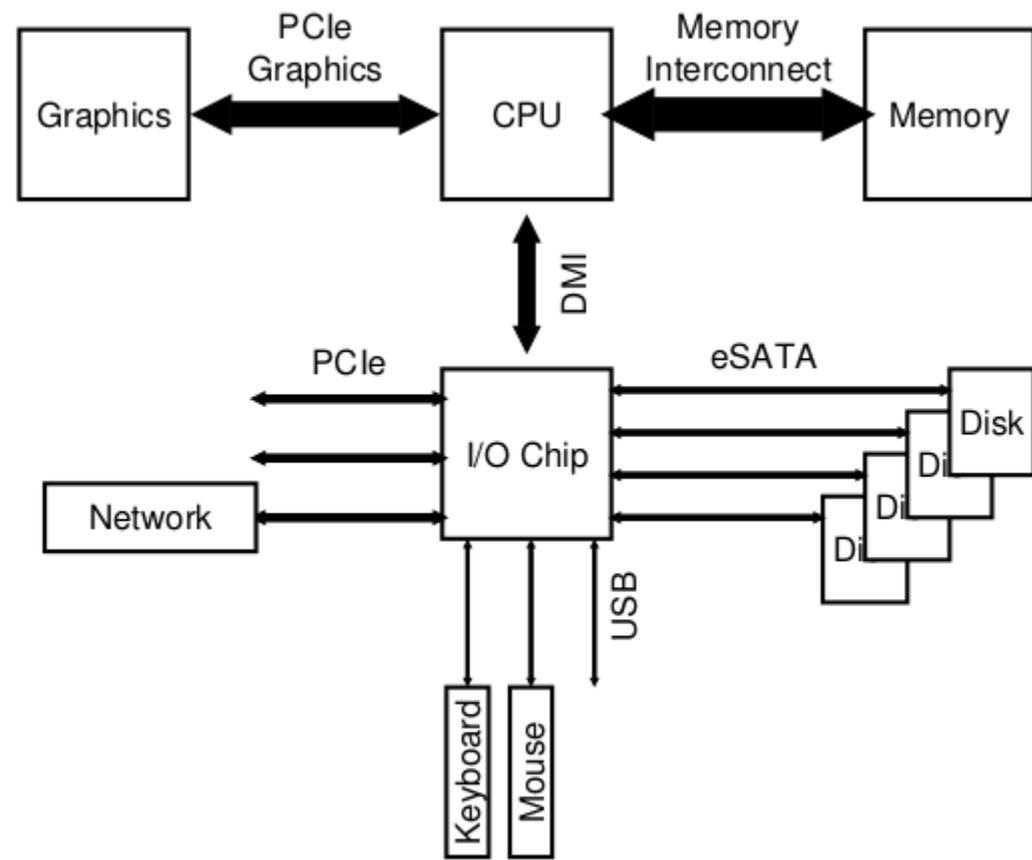
Figure 36.2: **Modern System Architecture**
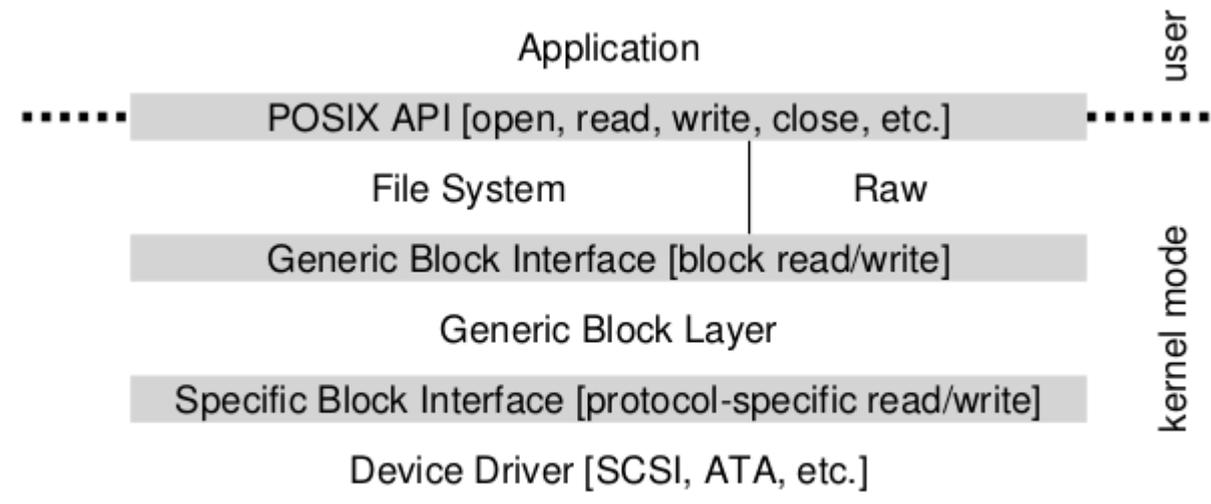
Figure 36.4: **The File System Stack**

2. PIO, Interrupts, DMA
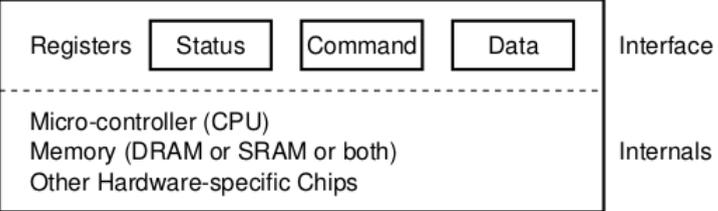
Figure 36.3: **A Canonical Device**

Programmed I/O and Polling:

```
While (STATUS == BUSY)
  ; // wait until device is not busy
Write data to DATA register
Write command to COMMAND register
  (starts the device and executes the command)
While (STATUS == BUSY)
  ; // wait until device is done with your request
```


Interrupts:
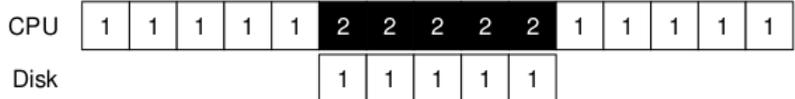
CPU Copying Data:
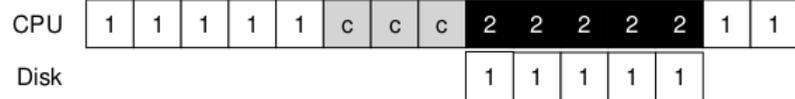
Direct Memory Access (DMA):
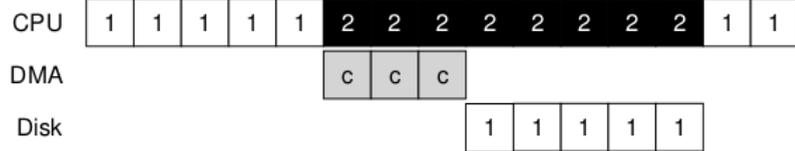
3. IDE Device Driver Example

```
static int ide_wait_ready() {
  while (((int r = inb(0x1f7)) & IDE_BSY) || !(r & IDE_DRDY))
    ;  // loop until drive isn't busy
}

static void ide_start_request(struct buf *b) {
  ide_wait_ready();
  outb(0x3f6, 0);                      // generate interrupt
  outb(0x1f2, 1);                      // how many sectors?
  outb(0x1f3, b->sector & 0xff);      // LBA goes here ...
  outb(0x1f4, (b->sector >> 8) & 0xff);   // ... and here
  outb(0x1f5, (b->sector >> 16) & 0xff);  // ... and here!
  outb(0x1f6, 0xe0 | ((b->dev&1)<<4) | ((b->sector>>24)&0x0f));
  if(b->flags & B_DIRTY){
    outb(0x1f7, IDE_CMD_WRITE);     // this is a WRITE
    outsl(0x1f0, b->data, 512/4);   // transfer data too!
  } else {
    outb(0x1f7, IDE_CMD_READ);      // this is a READ (no data)
  }
}

void ide_rw(struct buf *b) {
  acquire(&ide_lock);
  for (struct buf **pp = &ide_queue; *pp; pp=&(*pp)->qnext)
    ;                               // walk queue
  *pp = b;                          // add request to end
  if (ide_queue == b)               // if q is empty
    ide_start_request(b);           // send req to disk
  while ((b->flags & (B_VALID|B_DIRTY)) != B_VALID)
    sleep(b, &ide_lock);            // wait for completion
  release(&ide_lock);
}

void ide_intr() {
  struct buf *b;
  acquire(&ide_lock);
  if (!(b->flags & B_DIRTY) && ide_wait_ready() >= 0)
    insl(0x1f0, b->data, 512/4);    // if READ: get data
  b->flags |= B_VALID;
  b->flags &= ~B_DIRTY;
  wakeup(b);                        // wake waiting process
  if ((ide_queue = b->qnext) != 0)  // start next request
    ide_start_request(ide_queue);   // (if one exists)
  release(&ide_lock);
}
```

Figure 36.6: **The xv6 IDE Disk Driver (Simplified)**

```
Control Register:
  Address 0x3F6 = 0x08 (0000 1RE0): R=reset,
                       E=0 means "enable interrupt"

Command Block Registers:
  Address 0x1F0 = Data Port
  Address 0x1F1 = Error
  Address 0x1F2 = Sector Count
  Address 0x1F3 = LBA low byte
  Address 0x1F4 = LBA mid byte
  Address 0x1F5 = LBA hi  byte
  Address 0x1F6 = 1B1D TOP4LBA: B=LBA, D=drive
  Address 0x1F7 = Command/status

Status Register (Address 0x1F7):
     7      6     5      4     3     2     1      0
   BUSY  READY FAULT SEEK  DRQ   CORR IDDEX ERROR

Error Register (Address 0x1F1): (check when ERROR==1)
     7      6     5      4     3     2     1     0
   BBK     UNC   MC    IDNF  MCR   ABRT T0NF AMNF

  BBK  = Bad Block
  UNC  = Uncorrectable data error
  MC   = Media Changed
  IDNF = ID mark Not Found
  MCR  = Media Change Requested
  ABRT = Command aborted
  T0NF = Track 0 Not Found
  AMNF = Address Mark Not Found
```

Figure 36.5: **The IDE Interface**