

Driver layer

xv6-rev7

Carmi Merimovich

Tel-Aviv Academic College

January 20, 2017

IDE ports

out Ports

Port	Description
3f6	Interrupt enable
1f0	Data
1f2	Number of sectors to RW
1f3	Bits 0-7 of sector #r
1f4	bits 8-15 of sector #
1f6	bits 16-23 of sector #
1f6	bits 24-27 of sector # drive
1f7	Command

in Ports

Port	Description
1f7	Status

sector #/drive

	7	6	5	4	3	2	1	0
0x1f3 {								
0x1f4 {								
0x1f5 {								
0x1f6 {	1	1	1	d				

status port



B - Controller busy.

R - Drive ready.

D - Drive fault.

E - Error.

Loading data into the IDE buffer

1. Loop over longs:

```
long *a = (long *)b->data;  
for (int i = 0; i < 128; i++)  
    outl(0x1f0, a[i]);
```

2. Use the string instruction variety:

```
outsl(0x1f0, b->data, 128);
```

3. Loop over shorts:

```
short *a = (short *)b->data;  
for (int i = 0; i < 256; i++)  
    outw(0x1f0, a[i]);
```

Getting data from the IDE buffer

1. Loop over longs:

```
long *a = (long *)b->data;  
for (int i = 0; i < 128; i++)  
    a[i] = inl(0x1f0);
```

2. Use the string instruction variety:

```
insl(0x1f0, b->data, 128);
```

3. Loop over shorts:

```
short *a = (short *)b->data;  
for (int i = 0; i < 256; i++)  
    a[i] = inw(0x1f0);
```

idewait

- The IDE responds well to commands if it is ready.
- The IDE is deemed ready if it is not BUSY and it is DRDY.
- So idewait tight loops until the controller is ready.

idewait

```
3813 #define IDE_BSY 0x80
      #define IDE_DRDY 0x40
      #define IDE_DF 0x20
      #define IDE_ERR 0x01

3832 static int idewait(int checkerr) {
      int r;

      while(((r = inb(0x1f7)) &
              (IDE_BSY|IDE_DRDY)) != IDE_DRDY);
      if (checkerr && (r & (IDE_DF|IDE_ERR)) != 0)
          return -1;
      return 0;
}
```


idestart

```
3874 static void idestart(struct buf *b) {
    if (b == 0)
        panic("idestart");

    idewait(0);
    outb(0x3f6, 0); // generate interrupt
    outb(0x1f2, 1); // number of sectors
    outb(0x1f3, b->sector & 0xff);
    outb(0x1f4, (b->sector >> 8) & 0xff);
    outb(0x1f5, (b->sector >> 16) & 0xff);
    outb(0x1f6, 0xe0 | ((b->dev&1)<<4) |
                ((b->sector>>24)&0x0f));
    if (b->flags & B_DIRTY){
        outb(0x1f7, IDE_CMD_WRITE);
        outsl(0x1f0, b->data, 512/4);
    } else {
        outb(0x1f7, IDE_CMD_READ);
    }
}
```

iderw(b)

- A linked list of requested headed by idequeue is maintained.
- The buffer b is added at the tail of the linked list.
- If b is at the head of the queue, the controller is started on it.
- The process goes SLEEPING until the buffer is VALID and not DIRTY.

iderw

```
3953 void iderw(struct buf *b) {  
    struct buf **pp;  
    acquire(&idelock);  
    b->qnext = 0;  
    for(pp=&idequeue; *pp; pp=&(*pp)->qnext) ;  
    *pp = b;  
  
    if(idequeue == b)  
        idestart(b);  
  
    while((b->flags & (B_VALID|B_DIRTY)) != B_VALID){  
        sleep(b, &idelock);  
    }  
  
    release(&idelock);  
}
```

ideintr

3901

```
void ideintr(void) {  
    struct buf *b;  
    acquire(&idelock);  
    if((b = idequeue) == 0){  
        release(&idelock);  
        return;  
    }  
    idequeue = b->qnext;  
  
    if(!(b->flags & B_DIRTY) && idewait(1) >= 0)  
        insl(0x1f0, b->data, 512/4);  
  
    b->flags |= B_VALID;  
    b->flags &= ~B_DIRTY;  
    wakeup(b);  
  
    if(idequeue != 0) idestart(idequeue);  
    release(&idelock);  
}
```