Short Assignment – Devices and disks

Question 1: disk performance

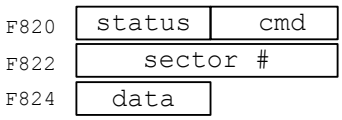
For this question, MB = 10^6 bytes

a) A disk rotates at 7200 RPM (120 rotations/sec) and can transfer 200MB/s of data from its outer track. How many bytes* of data does a single outer track hold?

b) Given an average seek time of 4ms, a rotational speed of 10000 RPM (166.66 rotations/sec), and an average transfer rate of 150 MB/s**, how long does a 65536-byte random read request take, in milliseconds, on average? What if transfer rate remains 150MB/s, but the average seek time is 8ms and rotational speed is 7200 RPM?

Question 2: simple disk driver

A slightly modified version of the really simple disk controller from HOSW is shown at the left; it supports reading and writing single 512-byte sectors from a very small disk (64K 512B sectors = 32MB), and has an associated interrupt. (see next page for details of operation)

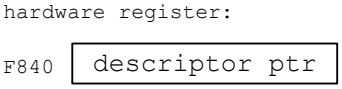


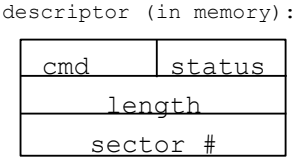
Provide pseudo-code for a simple interrupt-driven driver for this disk. You'll need to show logic for the following functions:

- `read(sector_num, bufptr, num_bytes)` - `num_bytes` is a multiple of 512
- `write(sector_num, bufptr, num_bytes)` - multiple of 512 again
- `disk_interrupt()` - function called for disk interrupt

Assume you have a mutex named `m` and a condition variable named `C`, - the `read` and `write` functions should call `C.wait(m)`, and `disk_interrupt` should call `C.signal()`

Question 3: complicated disk controller





Our imaginary machine has a DMA-based disk, controller, as well. You allocate a block of memory for a DMA descriptor, fill it out, and write its address to the descriptor pointer register. That signals the controller to perform its read or write operation, and your interrupt handler gets called when the I/O is done. Again, full details are on the next page.

alternately:

```
struct desc {
    int8 cmd;
    int8 status;
    int16 len;
    void* sector;
};
```

Provide pseudo-code for the `read`, `write`, and `disk_interrupt` functions. Again, you are given a mutex and condition variable for your read and write functions to use to wait for I/O to complete.

* Approximately. Assume the drive can skip from track to track with zero delay, which is not quite true.

** These numbers are about right for high-performance disks 15 years ago, and capacity drives were about 8ms/7200 RPM/150MB/s back then. Today's capacity drives are still 7200RPM and about 8ms seek time, but the max transfer rate is more like 200 to 300 MB/s because the bits are smaller.

The **simple disk controller** reads or writes exactly one 512-byte sector in response to a read or a write command.

Read:

1. write sector number to appropriate register
2. write CMD_READ (0x80) to the `cmd` register; in response the controller will:
   a) read and buffer the sector
   b) raise an interrupt, causing your handler to be called
3. [check that status=1, "no error"] - you can skip this
4. To get the buffered sector, software reads 512 bytes from the 8-bit data register

Write:

1. Write 512 8-bit bytes of data to the `data` register
2. Write sector number to sector# register
3. Write CMD_WRITE (0xC0) to the `cmd` register
4. When the transfer is completed, your interrupt handler will be called. [optional: check status=1]

The DMA disk controller can read or write multiple sectors:

1. Prepare a DMA descriptor in memory, with command (CMD_READ=0x80 or CMD_WRITE=0xC0), transfer length in bytes, 0 in the status field, and a pointer to the memory containing the data (WRITE) or where the data should be stored (READ)
2. Write the address of the DMA descriptor to the `descriptor_addr` register
3. When the transfer is done, your interrupt handler will be called. [optional: check status==1]