Questions:

1. Given the limited access to the system bus, which is a more appropriate I/O method for a keyboard: DMA or interrupts? Briefly explain why.

SOLUTION:
The answer is interrupts. The user keystrokes are not stored directly in main memory e.g. as bytes within a page of some process, which is what DMA does. Rather the keystrokes are data or instructions (command line) that must be handled by the kernel (and hence processed by CPU).

Some of you said that if continual typing on the keyboard were to interrupt the CPU then this would be inefficient. But that is incorrect. The CPU has billions ($10^9$) of clock cycles per second, whereas handling an interrupt might cost a few hundred clock cycles. We type maybe 5 keystrokes per second. If you were earning a billion dollars per year, you would not mind spending a few hundred on an important advisor.

Still not sure why DMA is wrong here? Ask yourself what DMA is for. Answer: DMA is used to transfer large amounts of information to and from main memory, while the CPU is occupied with something more important. (DMA is great for page faults, sending postscript files to the printer, sending audio files to a sound card, etc.) There would be no point in putting keyboard strokes in main memory, instead of having them on the I/O device controller.

2. The local memory of the hard disk controller is sometimes referred to as a disk cache, rather than a disk buffer. Why is the term "cache" suitable here? Hint: use the analogy of a "block" in your answer.

SOLUTION:
When reading a page from hard disk, the hard disk controller grabs neighboring pages as well. I mentioned this at the end of lecture 20 and, although I didn’t spend time on the details of the idea, it was a key idea. These neighboring physical pages often correspond to neighboring pages in the program’s (virtual) address space. Note this is different from the memory’s more basic role as a "buffer", namely as a place to hold a set of data before it can be moved somewhere else.

Loading multiple pages from disk when only one is requested is analogous to the CPU cache loading in a block of words from main memory when only a single word is requested. The goal is the same: take advantage of "locality of reference" – the fact that nearby addresses tend to be loaded at nearby times.

[Many of you wrote about how a cache gives fast access and you explained literally how blocks work in a cache. But this isn’t answering the question which was explicitly about the term cache vs buffer and the analogy of a block. We did not give any points if you just wrote how a hard disk works.]
3. In a daisy chaining interrupt request scheme, what is the role of the IACK signal from I/O controller \( n \) to I/O controller \( n + 1 \) ?

**SOLUTION:**
Here is what we were looking for: When IACK goes from 0 to 1, it indicates that (1) the *CPU is acknowledging the interrupt and that the bus is free* and that (2) *no higher priority devices have made an interrupt request.* When it goes from 1 to 0, it indicates that permission to write on the bus has been taken away.

Defining a grading scheme was not straightforward for this one. Generally we gave points for pieces of the above. But if no mention was made of the system bus, then we tended not to give points.

4. Suppose the one-byte ASCII codes, \((10100011)_2\) and \((00100100)_2\) were sent as part of a message on a *serial bus*. Give an example of the sequence of bits that would be read from the bus. Indicate which bit is read first and label any other bits that are sent.

**SOLUTION:**
The underlined bits below show a sequence of 1’s followed by a 0 (the start bit). The non-underlined bits are the ASCII codes above. The first 1 bit following an 8 bit ASCII code is called the stop bit.

\[ ...111110 \ 10100011 \ 111...0 \ 00100100 \ 111... \]

I put a gap between the ASCII codes and the start and stop bits to make it easier to read. In a real situation there is no gap.

Some students mentioned about the parity bit. Its great that you learned that, but it doesn’t help for this question. Others did not indicate the start or stop bit, which was the main point of the question. You cannot communicate information on a serial line without such bits.