Questions

1. When servicing an interrupt, the registers of the current process must be saved. Where are they saved?

2. Recall the five stages of the MIPS pipeline. Other than interrupts (which can occur at any time), which exceptions can occur at each stage of the pipeline?

3. Give an example of an I/O device other than the hard disk for which DMA could be used.

4. Suppose there is a cache miss on a `sw` instruction. Describe the steps that are taken, assuming a write-back policy. Specify when the system bus is used, and how it is used.

5. In MIPS, what exception (*i.e.* unexpected event) occurs if the following pair of instructions are executed ?

```
lui    $ra, 0xffff      # load upper immediate
jr     $ra              # jump to address contained in $ra
```

   How might a processor (CPU) be designed to automatically detect such an exception ?

6. On some computers, it is fine to take out your USB stick from your computer, whereas on others you have to go through a 'safely eject hardware' step. Why?

Solutions

1. It depends what you mean by "where" – virtual or physical? If you mean "virtual", then the answer is simply that register values are saved in the kernel's half of the address space $0x80000000$ and above. If you mean "physica", then the answer is 'in RAM', in particular, in the part of RAM that is dedicated to the kernel. (This may or may not be paged. I didn't discuss paging the kernel region in class, but it is possible.)

   Either way, think of a data structure for a process which includes the page table as well as other admin info for the process (such as how long it has been running, what its priority is, etc), and register values if that process has been temporarily halted.

   Finally, you might have said "the stack". That's not correct.

2. Some examples are

   - IF - page fault, bad address (e.g. user tries fetching from kernel or data area)
   - ID - non-valid op code (some opcodes are not used)
   - ALU - arithmetic exception (e.g. integer overflow)
   - MEM - page fault, bad address (see IF)
   - WB - writing into an illegal register number e.g. $0

3. The printer is a great example. One might have a PostScript (or PDF) file in main memory and want to print it. Pages would need to be sent to the printer but there is no need for the CPU to be heavily involved here. The printer's driver (a program running on CPU) could issue instructions to the printer's controller which could then take over the bus and arrange for the data to be sent from main memory to the printer.

4. First, realize that the word must be in main memory (not on the hard disk). Why? Because if we are indexing in the cache then we have passed the TLB stage which means that the physical address of the page holding the word cannot be on the hard disk.

   Now there are two cases: the dirty bit for the cache entry is on, or the dirty bit is off. If the dirty bit is on, then the block needs to be written back to main memory before the new block is read in. The CPU puts the block of data on the data bus, it puts the address of the block in main memory on the address bus, and it puts a control signal (write-memory) on the control bus. Main memory then reads the block. (This may take a few bus clock cycles, depending on how many words are in the block.) If the dirty bit is off, then the data in the block is identical to the corresponding block in main memory and so there is no need to write it back.

   The CPU next needs to bring a block from the main memory to the cache. This requires that the CPU puts the physical address of the block in main memory on the address bus, and it puts a control signal (write-memory) on the control bus. These two signals are read by main memory which puts the requested block on the data bus. The block is read by the CPU and written into the cache.

   Finally, the data from the register (specified in the `sw` instruction) is written into the cache entry. The dirty bit of the cache entry is set to 1.

5. You probably answered that a user program jumped to the kernel, which is not allowed. This is correct, however, the answer is a bit more subtle. The exception is that the PC would contain a memory mapped I/O address. (The upper 16 bits being on corresponds to the upper 64K bytes in MIPS memory, which is where MIPS does memory mapped I/O.) This is an error since it would mean that the current instruction is found at an I/O register, which makes no sense.

   BTW, no, I am not expecting you to remember the above! I'm just using this as a reminder of what memory mapped I/O is in case you forgot.

   Also, such an error could be detected easily in hardware. At the beginning of the fetch sequence, there could be a control signal computed that checks if the upper 16 bits of the PC are all ON. If so, then an exception occurs.

6. The key difference is whether the computer is using a 'write-through' or 'write-back' scheme to maintain consistency between pages on the USB drive and pages in RAM. (This is the same idea as we saw with caches and RAM.) Think of a file such as Word document or Excel spreadsheet. When you are working on this document, there is the document itself on the hard drive (which is what you mean when you refer to the 'file') and there are copies of pages (perhaps all pages of the file) that are in RAM and that are being edited.

   A 'write-through' scheme for RAM and the hard drive would mean that whenever a page in RAM is written to (by the CPU) the corresponding page on the hard drive would be updated so that there is consistency between these pages. A 'write back' scheme would only modify the pages on the hard drive when it is necessary. An example of when it is necessary would be if there were a page fault and the modified page from the file needed to be moved out of RAM and onto the hard disk. Another example would be if the user wants to take the USB drive out of the computer.

   If a 'write through' scheme is used to maintain consistency between main memory and the USB drive, then the user can take the USB drive out whenever they want (as long as the drive is not being written to at that time, of course, and as long as there are no voltage fluctuation issues with ripping out the drive – we are ignoring the latter). If a 'write back' scheme is used, however, then the user needs to first click on 'safely remove hardware' which will cause the OS to write back the page from RAM to the USB to maintain consistency, and once that is done the user can take out the drive.