OSI Week10: I/O and device driver

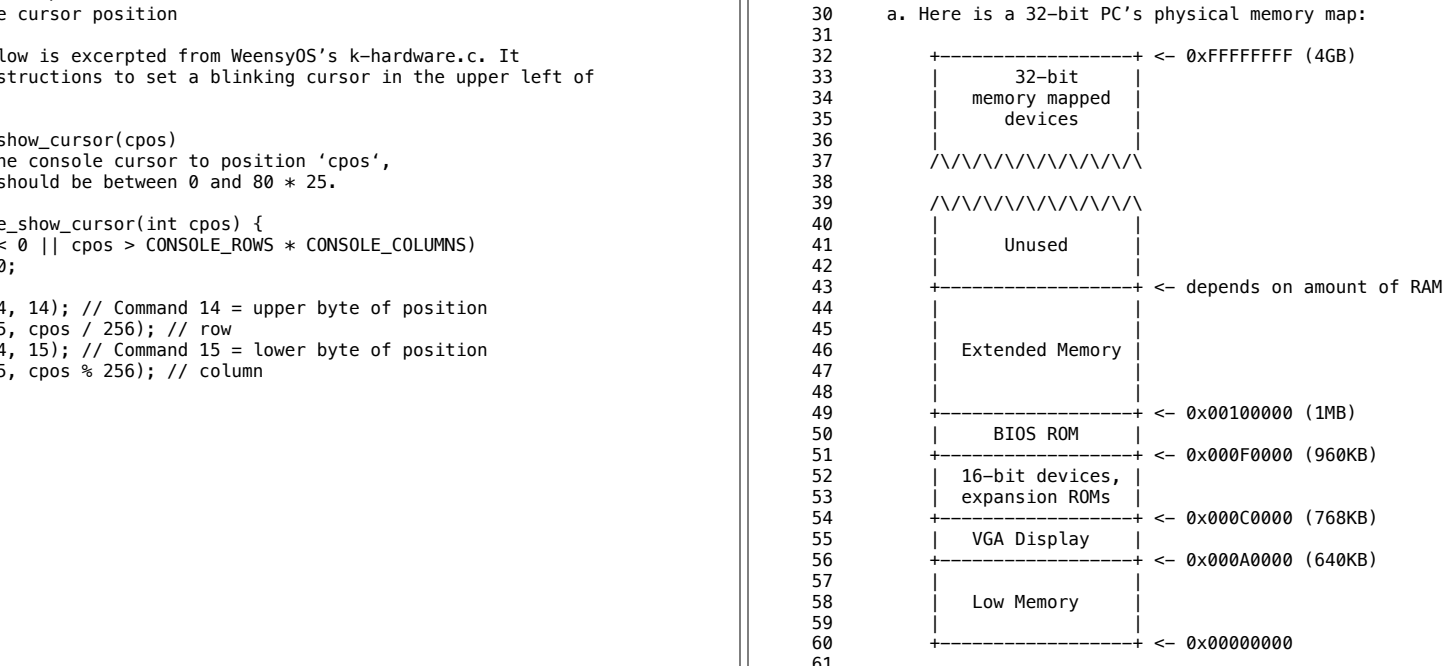
1. An example of I/O instructions:
   Setting the cursor position

  The code below is excerpted from WeensyOS's k-hardware.c. It
  uses I/O instructions to set a blinking cursor in the upper left of
  the screen.

```
// console_show_cursor(cpos)
//   Move the console cursor to position 'cpos',
//   which should be between 0 and 80 * 25.

void console_show_cursor(int cpos) {
  if (cpos < 0 || cpos > CONSOLE_ROWS * CONSOLE_COLUMNS)
    cpos = 0;

  outb(0x3D4, 14); // Command 14 = upper byte of position
  outb(0x3D5, cpos / 256); // row
  outb(0x3D4, 15); // Command 15 = lower byte of position
  outb(0x3D5, cpos % 256); // column

}
```


2. Memory-mapped I/O

  a. Here is a 32-bit PC's physical memory map:

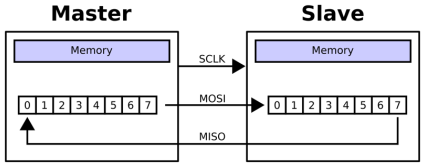


  [Credit to Frans Kaashoek, Robert Morris, and
  Nickolai Zeldovich for this picture]

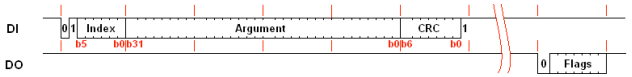
```
65
66      b. Loads and stores to the device memory "go to hardware".
67
68        An example is in the console printing code from WeensyOS.
69        Here is an excerpt from link/shared.ld:
70
71        /* Compare the address below to the map above. */
72        PROVIDE(console = 0xB8000);
73
74        This is an excerpt from lib.c; notice how it uses the address
75        "console":
76
77        /*
78        * prints a character to the console at the specified
79        * cursor position in the specified color.
80        * Question: what is going on in the check
81        * if (c == '\n')
82        * ?
83        * Hint: '\n' is "C" for "newline" (the user pressed enter).
84        */
85        static void console_putc(printer* p, unsigned char c, int color) {
86          console_printer* cp = (console_printer*) p;
87          if (cp->cursor >= console + CONSOLE_ROWS * CONSOLE_COLUMNS) {
88            cp->cursor = console;
89          }
90          if (c == '\n') {
91            int pos = (cp->cursor - console) % 80;
92            for (; pos != 80; pos++) {
93              *cp->cursor++ = ' ' | color;
94            }
95          } else {
96            *cp->cursor++ = c | color;
97          }
98        }
```

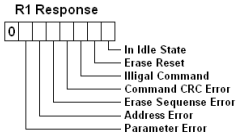

3. SPI basics

- SPI architecture

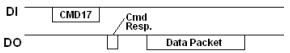


- Command and response

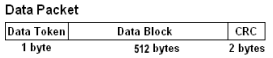




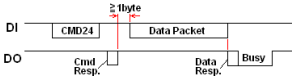
- Single-block read



- Data packet



- Single-block write



- Data response