Scott Fenton

MP1

Discussion questions

1. I tested my code by uploading it incrementally and testing each small part until every command worked as intended. I verified that it worked by testing it against the commands used by the tutor system.

2. When I call md for an address that does not correspond to a physical memory address, the program on the SAPC calls an “Exception 14 at EIP=0010048f: page fault” and exits the running program. This happened when I called md 400000, but when I called md 100000000000, the memory address went back to 0x000000. When I call md on ROM and tutor code, it works as normally. It works differently on SAPC and UNIX because in the Unix environment the memory is protected, so I cannot set a pointer to a memory address that is outside of the program scope.

3.a. The address of xyz in memory is 0x10223c with value of 06 00 00 00.

3.b. The address of pxyz in memory is 0x102240 with value 3c 22 10 00. This is the same value that is beside xyz at memory address 0x10223c.

3.c. The address of the ‘m’ is 0x10042a, and ‘d’ is 0x10042b.

3.d. You can change the tutor prompt the same way. If you change value 0x100211 you can change the first letter of the PC-tutor to whatever you want.

4. If I go to 0x100100 I can find the start of my program code. At 0x100200 I start to see fragments of code that are printed to the console, like “No such command”, and “MD”, “MS”, and others. The data variables are toward the end of my program memory.

5. If I change the code at Address 0x100765 on the tutor SAPC machine, then the program automatically crashes, and exits to the underlying tutor program. On the Unix machine I can only change the memory contents that are in the running programs memory, so if I try to change the memory contents my program crashes with a segmentation fault error.

6. The stack pointer ESP is at 0x3ffff0, at the end of the program memory. If I try to go any further, then I get an Exception 14: page fault error, and cannot go any further.

7. I looked at the memory addresses after my program memory, and they are all initialized to 00. I found it interesting that you can compile the c program on Unix but run into problems if the code is not formatted a certain way when compiling it to lnx format. For example, if you assign int x = 0; after a for loop, it will compile in c but will throw an error when compiling into lnx format. Changing certain memory addresses in tutor will change what shows up as console text.

8. I learned a lot more about pointers from this program, and some of the idiosyncrasies from working on a low level system. The hardest parts were when it compiled correctly, but the data was not manipulated the way you intended, and having to reload it to the VM. It was surprising to find out how easy it is to manipulate the program data if given the access.