Recitation 13
PA6 Help

Today we're going to inspect how the code we've given you for PA6 models the LC3, as well as revisit a few debugger features that will make more sense now that we've seen stacks and pointers.

(The file below includes a few source files which were already compiled into the PA6 file lc3sim.a for you, but having them allows you to see more in the debugger. If you want to get the additional files in your PA6 directory for later use, you can run "`tar -xf ~cs270/lc3tools/simSrc.tar`")

```
wget http://www.cs.colostate.edu/~cs270/.Fall13/recitations/R13/R13.tar.gz
tar -xzf R13.tar.gz
cd R13
ddd mysim
```

(note: ddd was originally written in an era before current GUI behavior became a common convention. As a result, you must hover the mouse cursor over whatever field you wish to type into)

File->Open Source... "logic.c"

Find the function `logic_read_memory`, **right click** on it and choose "**Break at logic_read_memory**"

Click "**Run**" in the DDD Command Tool (the separate window with only buttons)

It should stop at the breakpoint.

Now some setup stuff to have the debugger interpret the stack memory for us (including stack/local variables and arguments)

In the ddd GUI, do:

Status->Backtrace... (this will bring up a window showing you the stack)
Data->Display Local Variables
Data->Display Arguments

You should see one local variable and one argument in the display area. You should also see in the backtrace (stack) window a series of function calls starting with `main` and leading to `logic_read_memory`. If you click on each stack frame (activation record) the source code window will show you what the PC was in that frame when it called the next function in the stack.

Now click "**Step**" (in the Command Tool) until you step *into* the function `hardware_load_MAR`. Notice what happened to the backtrace window and the Locals and Args displays. Now examine

that function and observe that it assigns to a variable, `reg_MAR` (it is declared as static (i.e. limited to that file), so your PA6 code *must* use the `hardware_load_MAR` function to change it). **Right click** on `reg_MAR` and choose "**Display reg_MAR**". Do the same for lc3_BUS. **Double click** on the hex value in the lc3_BUS display to see what it points to.

Scroll up in this file (hardware.c) and find the declarations for all the other variables used to model LC3 registers. **Right click** and choose display for each of the following:

lc3_registers
reg_PC
reg_IR
reg_PSR
reg_MDR

Continue stepping until you reach the end of `logic_read_memory` (but once you step inside the function `hardware_memory_enable`, click **Finish** to tell it to run to the completion of the current function).

Now do **Source->Breakpoints...** and **disable** the `logic_read_memory` brakepoint.

Next scroll down in `logic.c` (the main DDD window should say logic.c in the title bar) and set a breakpoint on `logic_fetch_instruction`.

Click "**Cont**" (in the Command Tool) to continue execution till the next breakpoint. You'll notice lc3sim does in fact continue with normal operations and presents an lc3sim prompt inside the ddd window. Type `step` at the `(lc3sim)` prompt to tell the running lc3sim simulation to run one instruction. You should see that lc3sim is now stopped at the breakpoint. **Double click** on the hexadecimal pointer value shown for the "`inst`" argument in the Args display.

It's impractical to display the 65,536 element lc3_memory array in the display area the same way we can with the 8 element lc3_register file, so we'll use the gdb prompt instead. Type:

```
p lc3_memory[5]
```

at the (gdb) prompt in the bottom window to see what's at address 5.

Show your instructor your simulation window or File->Print Graph... (choose to print to a .ps file) and email them this file.