CSCI 2271 Computer Systems
Class 4: 1/27/16

1. How the Stack is Used

- The reason for the weird stack layout is that it is easy to implement.
  - As an example, consider the handout.
  - It has two columns.
  - The first column shows some C code that calls a function.
  - The second column shows the corresponding machine-like code that gets generated.
- The handout has two pages.
  - The first page shows how a void function is handled.
  - The second page considers a non-void function.
- Let's first look at the code for *main*, without the call to *g*. Issues:
  - Allocation of space for the local vars.
  - Accessing the contents of a local var.
  - Deallocating space for the local vars.
- Now consider the function call. Issues:
  - Passing the params.
  - Saving the address to return to.
  - Deallocating the params.
- What the function g does:
  - pushes the current contents of the base register on the stack;
  - sets the base register to be the current top of stack;
  - increases the stack to contain space for local variables;
  - executes the code for g.
- When the code for g has finished, it undoes it actions on the stack:
  - deallocates the local variable.
  - pops the stack, making the previous frame current.
  - pops the stack to find the address of the next instruction to execute.
- Draw a picture of these events.
  - Note that the caller is responsible for allocating the top half of the stack frame, and the callee is responsible to the bottom half.
- Also note that the system does not initialize the values of the local variables when a stack frame is created.
  - Instead, initialization must be performed by the code for that function.
  - That is why, if you don't initialize a local variable in your code, its value may be garbage.
-