CSCI 2271 Computer Systems

Class 5: 1/29/16

1. Go Over Quiz 1

- Issue: variable allocation
  - Local variables are allocated in the stack segment. Global variables are allocated in the data segment.
  - Local variables are allocated by size, with the largest sizes allocated first. Allocation proceeds from high addresses to low ones.
  - I think it is easier to calculate addresses using hexadecimal, but if you want to do everything in base 10 it is ok with me.
- Issue: *printf* format designators
  - When you say "%4.2f", the 4 is the *width* and the 2 is the *precision*.
  - The width is the number of characters that will be used to display the value.
  - The precision of a floating point number is the number of digits to the right of the decimal point. If necessary, either add zeroes to the end of the number or truncate the number.
  - The precision of an integer is the total number of digits to display. If necessary, add zeroes to the beginning of the number.
- Issue: chars are numbers
  - Like Java, a char is a number. It is just a short integer.
  - Unlike Java, the compiler lets you do any numeric calculation you want on a char.
- Issue: the definition of true and false
  - Early in the semester, I said that C uses 1 to mean true and 0 to mean false.
  - That isn't quite true.
  - Actually, C uses 0 to mean false and anything else to mean true.
  - Sometimes this fact is useful.

2. The Structure of a Stack Frame

- A stack frame contains four kinds of thing, in this order:
  - parameter values
  - the address of the instruction to branch to when this function exits.
  - the address the the base register should contain when the function exits.
  - the local variables.
- Suppose for example that function f wants to call g. Here is how the stack frame for g gets set up.
- Before f branches to the code for g, it sets up the first two parts of the stack frame:
  - It computes the parameter values and pushes them on the stack.

    - Then it pushes the instruction to branch to on the stack.
- When g starts, it finishes setting up the stack frame:
    - It pushes the current value of the base register on the stack.
    - It allocates space for g's local variables.
- When g finishes, it does the following:
    - It deallocates the local variables.
    - It pops the stack (which is the value of the base register when f was active) and uses that value to restore the base register.
    - it pops the stack (which is the address of the next instruction of f to execute) and branches to that address.
- When f starts up again, it deallocates g's parameter values from the stack.
- Note that both f and g collaborate to create g's stack frame.

3. Address Values

- The stack frame contains two kinds of value.
    - Numbers, such as integers and floats.
    - Addresses of memory locations.
- The base pointer and return address values are addresses.
- The values of parameters and local variables depend on the type of the corresponding variables in the C program.
    - At the moment we only know about numeric types, so these values will be numbers.
    - Later, we shall see how pointers can also be saved here.
- A value that stores an address is called a *pointer*.
- What is the size of a pointer?
    - It depends on the computer hardware and the operating system.
- The rule is that the more bytes the OS uses to store pointers, the more memory it can access.
    - For example, suppose pointers are 1 byte (the same size as a character), where a byte is 8 bits.
    - There are $2^8 = 256$ possible values you can represent using 8 bits, which means that a 1-byte pointer can access at most 256 possible addresses.
    - Similarly, a 2-byte pointer can access $2^{16} = 256^2 = 65,536$ possible addresses. Still not very many.
    - A 4-byte pointer can access $2^{32} = 65536^2 = 4,294,967,296$ possible addresses. This is about 4 GB.
- Our Linux system uses 4-byte addresses.
    - Usually, the pointer size is in bits.
    - So our Linux system is called a *32-bit* OS.
- Modern OS's use 64-bit (or 8-byte) pointers.
    - This allows a program to reference a gigantic number of addresses.
- For this course, we shall assume that all pointers are 4-bytes.