Welcome back to the third lecture about stack memory.

First, let us review what we have learned so far. The stack memory can have three things: The first is the return location. This is the line number after the function call. Each function call has one and only one return location. The second is argument (or arguments). This is the input to the function. The third is the local variable (or variables).

The stack memory may store another type of information. Before I explain that, I am going to describe how memory is organized inside computers.

Computer memory is organized in address-value pairs. Each row has an address and a value is stored at that address. You can think of the addresses in computer memory as the addresses of houses. The values are the names of the families living in the houses. For example, along a street. The first house lives the family of Johnson. The second address belongs to the family of Smith. The third address is for the family of Taylor.

A computer’s memory is organized in a similar way. Each unit of memory has an address and at that address stores some information. The operating system guarantees that two pieces of information have different addresses. Also, the address is never zero.

Here are some rules about memory addresses. First, operating systems assign addresses. Moreover, two different pieces of information must reside in two different addresses. This is guaranteed by operating systems. As a programmer, you have no control of the addresses given to your programs. When operating systems assign addresses, one special address is never used and it is zero. This address is so special, C programs use a special symbol N U L L (or null) for this address. We will see how this symbol can be used in later lectures.

The fourth type of information that can be stored in the stack memory is called the value address.

So far, I have been ignoring the frame of f1. Since f1 is also a function, it follows the same rules. Its frame contains the return location of the line number after f1 is called. The argument x has a value assigned by the function where f1 is called. Function f1 has a local variable "Ae". Function f1 calls function f3 and the value returned by function f3 is written to "Ae". The address of "Ae" is stored in the frame of f3.

In this example, I use 100 as the smallest address. As I mentioned earlier, programmers have no control of addresses and two pieces of information always have different addresses. For simplicity, I use 100, 101, 102, etc. for the addresses. The address of local variable "Ae" is 102. When function f3 finishes and returns an integer, this value integer is written to f1’s local variable "Ae". The address of "Ae" is 102. This is the value address. Earlier, I said computer memory is organized as address-value pairs. However, this slide does not show address-value pairs. The next slide explains how to map the concept to address-value pairs.

This slide shows how memory is actually organized.

Everything in the memory has an address. In this particular example, the local variable "Ae" inside function f1 has address 102. When f3 is called, the value address for "Ae" is 102 and it is stored in the stack memory.

Please notice how I present the stack memory. The first column is the frame. Each function has its own frame. This example does not show the caller of f1; thus, the return location is marked by a question mark. Also, I am not marking the value of "x" here. Instead, let’s focus on the frame of f3 in this example. Function f3 is called by f1 at line 4. Thus, the return location is line 5. The value returned from f3 is written to a so the value address is the address of "Ae" and it is 102. Please notice that the value address is stored in the stack memory and space is needed for storing this information. The value address is stored at address 104. Do not mix the value and the address. The value address is 102 and the address is stored at location 104.

The previous slide shows the four columns in the stack memory. In fact, the frames and the symbols are for humans only. Computers do not know frames and symbols. Computers know only addresses and values.

Even though computers do not know frames and symbols, it is helpful to include them when we draw the stack memory because the information help us keep track which function and which symbol we are discussing.

I have explained the four things that may be stored in stack memory. They are return locations, the arguments, local variables, and value addresses.

In the next lecture, we will see some examples from real C programs.