This lecture explains the concept of memory inside computers.

Let me start with a little history about programming languages. When C was developed, computers were expensive and slow. The designers of C language took the following philosophy

A C program does exactly what the program says, and nothing more.

What does this mean? Many things. C does not initialize variables for you. If your program creates a variable and it has not been initialized, its value can be anything. You must not assume uninitialized variables are zeros.

Some students assume that uninitialized variables are zero. This is incorrect.

C programs do not collect garbage. What is garbage in a computer program? When a computer program runs, it often needs to store data. If the data is no longer needed, it becomes “garbage”. Some programming languages (such as Java) automatically reclaim the space used for the data. This is called “garbage collection”.

C does not collect garbage for several reasons. The most important reason is that determining which pieces of data are no longer needed is a difficult problem and making the decision can slow down the program noticeably. When you write C programs, you are responsible releasing the memory used for storing data.

Also, C programs do not check errors because checking errors slows down programs. Programmers are responsible ensuring the correctness of their programs.

Memory is an important part of any computer. What is memory for inside a computer? How do programs use memory? Imagine that you are solving a complex problem. You need some reference books. This would be considered “read-only” memory. Read-only memory stores important information about the computers. The content in the memory cannot be erased or modified. I will not talk more about read-only memory in this series of lectures.

You also need scratch paper to record the steps and intermediate results. This would be equivalent to read-write memory. In early days, computer memory was extremely expensive. As a result, memory management is a very important part in writing computer programs. To be a good programmers, you need to know how to manage memory.

Computers are much cheaper now and your mobile phone has more memory than a supercomputer thirty years ago. Does that mean memory management is no longer important?

To the contrary, memory management is still important. In fact, some people think memory management is even more important today because the amounts of data grow very fast. Many sensors can capture data from all types of systems. Thus, memory management is still an important problem.

You probably have not seen this. This is called core memory. It is about the size of today’s mobile phone. However, a mobile phone can store more than ten million times more data than the core memory.

Inside a computer, memory is divided into two parts. The first is called “user space” used by ordinary programs. The other part is called “kernel space” and only the operating system can use. I will not talk about the kernel space any further in this series of lectures.

The memory in the user space is further divided into three parts: stack memory, heap memory, and program memory.

The stack memory follows a special rule called “first in, last out”, or “last in, first out”. The heap memory is more flexible. A program can allocate or release memory from the heap. The third part stores computer programs.

The program memory is managed by operating systems and will not be further discussed in this class.

Let me start by talking about the stack memory. It is one part of the memory. It is called stack because its behavior is like a stack. I will explain what stack means in the next slide. Stack memory is not directly controlled by your program. Instead, stack memory is controlled by compilers and operating systems.

What is a stack?

Stack means what comes first leaves last. You use this concept everyday. You put on socks before you put on shoes. You take off the shoes before you take off the socks. The socks come first but leave later.

You put on a shirt before you put on a jacket. You take off the jacket before you take off the shirt. Again, the shirt comes first and leaves later.

When put books on the top of a pile, the last book you put on is the first book you remove.

How is stack memory used in a computer program? Let’s consider this extremely simple program with three functions: f1, f2, and f3. f1 calls f2 and f2 calls f3. Imagine that the program starts at function f1. The program will then call f2 and then f3. When f3 finishes, the program continues from f2, not f1.

To be more precise, when the program executes function f1, there is one function, f1, stored in the stack memory.

Then, the program enters function f2. Now, the stack memory has f1 and f2. By convention, the newest item is put on the top of the stack, in the same way as a pile of books. When an item is added to a stack, this item is pushed to the top of the stack.

When function f3 finishes, the program continues f2 at the location after calling f3.

When function f2 finishes, the top of the stack memory is taken away. The stack memory has only f1 now and the program continues from the location after calling f2. When an item is removed from the top of a stack, this item is popped from the stack.

That’s the idea of the stack memory. The next lecture will provide more details.