Gmail by Google

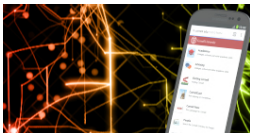
Karen West <karenwest15@gmail.com>

[The Computing Technology Inside Your Smartphone] Overview of week 6

The Computing Technology Inside Your Smartphone <ENGRI1210x-no-reply@courseupdates.edx.org> Tue, Apr 14, 2015 at 12:47 PM
To: KarenWest15@gmail.com

Connect with edX:

The Computing Technology Inside Your Smartphone



This week we reach the top of the stack! The topic for this week is programming, with an emphasis on the levels of programming and how programs get transformed into the zeros and ones that the hardware understands. After this week, we'll have the full picture of the stack, from transistors to applications. We'll also achieve a second major milestone by completing our LC-3 Lite design.

We have already seen programs as binary numbers (machine code) stored in memory; this is how they appear to the hardware. However, writing programs in machine code is tedious and error-prone. *Assembly language* is a human readable form of machine code that permits using symbols, such as ADD, in place of numbers, such as 0001 (the LC-3 opcode for ADD). It's certainly more intuitive to express, "Add the contents of R2 and R3 and put the result in R1" as ADD R1,R2,R3 rather than 0001001010000011! Assembly language also permits using labels to mark locations so that we don't have to calculate offsets for branches and PC-relative addressing mode.

A piece of system software called the *assembler* does the work of calculating those offsets and translating assembly language statements into machine code. A second piece of system software called a *linker* then combines multiple machine code modules (that may have been written by different programmers) to form the *executable* program that is stored on our smartphones. When the program starts, a third piece of system software called the *loader* transfers it into memory. We'll create an assembly language version of our character counting program, and then, like an assembler, translate a block of it into the machine code that the hardware understands.

We'll then move up to high-level languages (HLLs), in particular, the C programming language. HLLs abstract away the details of the ISA; we no longer need be concerned with registers, addressing modes, and the like. This permits us to be even more productive by using expressions such as a = b + c without

concern for how to retrieve those variables from memory and the registers they occupy. A piece of system software called the *compiler* does all that work for us by translating our HLL statements into machine code. Since HLL programs are not ISA-specific, they are *portable* in that they can be compiled for multiple ISAs, such as ARM and Intel. HLL programs can also be *interpreted* rather than compiled and we'll discuss the differences. Finally, we'll act like a compiler by translating HLL code into LC-3 assembly language (which could then be translated into machine code) to give you the full picture of the stack, from application programs to transistors.

We will also complete our LC-3 Lite design. First, we'll combine some components and then construct and test the complete design. After we cover assembly language, you'll write programs and run them on your hardware.

Having completed the stack, we'll be prepared to cover performance in the final two weeks of the course, where we'll learn the techniques that computer architects use to speed up our programs. Then we'll revisit the Snapdragon SoC, armed with the knowledge to delve into the details of the Krait processor.

As always, we are here to answer your questions. Be sure to use #SmartphoneMOOC when tweeting about the course.

On behalf of the course staff, good luck and have fun!

Dave

Our mailing address is:
edX
141 Portland Street
Cambridge, MA, USA 02139