Help

In this class we will focus our software development in C. However, the process described in this section applies to both assembly and C. Either the ARM Keil™ **uVision®** or the Texas Instruments **Code Composer Studio™ (CCStudio)** integrated development environment (IDE) can be used to develop software for the Texas Instruments microcontrollers. Both include an editor, assembler, compiler, and simulator. Furthermore, both can be used to download and debug software on a real microcontroller. Either way, the entire development process is contained in one application, as shown in Figure 2.22. In this course, we will use ARM Keil™ **uVision.**

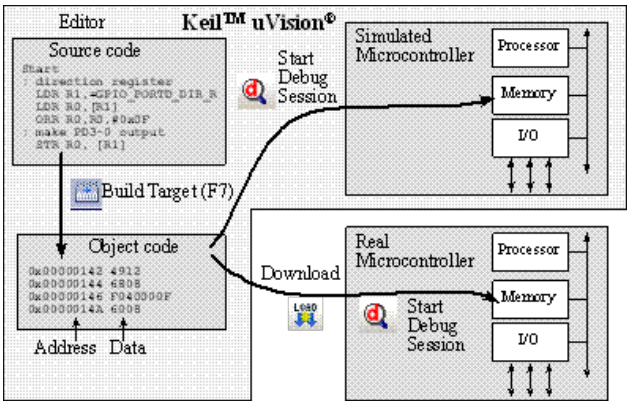

Figure 2.22. Assembly language or C development process.

To develop software, we first use an **editor** to create our **source code**. Source code contains specific set of sequential commands in human-readable-form. Next, we use an **assembler** or **compiler** to translate our source code into object code. On ARM Keil™ uVision® we compile/assemble by executing the command **Project->Build Target** (short cut F7). Object code or machine instructions contains these same commands in machine-readable-form. Most assembly source code is one-to-one with the object code that is executed by the computer. For example, when programming in a high level language like C or Java, one line of a program can translate into several machine instructions. In contrast, one line of assembly code usually translates to exactly one machine instruction. The assembler/compiler may also produce a **listing file**, which is a human-readable output showing the addresses and object code that correspond to each line of the source program. The **target** specifies the platform on which we will be running the object code. When testing software with the simulator, we choose the **Simulator** as the target. When simulating, there is no need to download, we simply launch the simulator by executing the **Debug->Start Debug Session** command. The simulator is an easy and inexpensive way to get started on a project. However, its usefulness will diminish as the I/O becomes more complex.

In a real system, we choose the real microcontroller via its JTAG debugger as the target. In this way the object code is downloaded into the EEPROM of the microcontroller. Most microcontrollers contain built-in features that assist in programming their EEPROM. In particular, we will use the JTAG debugger connected via a USB cable to download and

debug programs. The JTAG is both a **loader** and a **debugger**. We program the EEPROM by executing the **Flash->Download** command. After downloading we can start the system by hitting the reset button on the board or we can debug it by executing **Debug->Start Debug Session** command in the uVision® IDE.

In contrast, the loader on a general purpose computer typically reads the object code from a file on a hard drive or CD and stores the code in RAM. When the program is run, instructions are fetched from RAM. Since RAM is volatile, the programs on a general purpose computer must be loaded each time the system is powered up.

For embedded systems, we typically perform initial testing on a simulator. The process for developing applications on real hardware is identical except the target is switched from a simulated microcontroller to the real microcontroller. It is best to have a programming reference manual handy when writing assembly language. These four reference manuals for the Cortex M4 processor are available as pdf files, download them to your computer for reference.

CortexM_InstructionSet.pdf (/c4x/UTAustinX/UT.6.01x/asset/CortexM_InstructionSet.pdf) Instruction Set Reference Manual

CortexM4_TRM_r0p1.pdf (/c4x/UTAustinX/UT.6.01x/asset/CortexM4_TRM_r0p1.pdf) Cortex-M4 Technical Reference Manual

LaunchPadUsersManual.pdf (/c4x/UTAustinX/UT.6.01x/asset/LaunchPadUsersManual.pdf) LaunchPad Manual

tm4c123gh6pm.pdf (/c4x/UTAustinX/UT.6.01x/asset/tm4c123gh6pm.pdf) Data Sheet for the TM4C123 microcontroller

Help

A description of each instruction can also be found by searching the Contents page of the help engine included with the ARM Keil™ uVision® . There are a lot of settings required to create a software project from scratch. I strongly suggest those new to the process first run lots of existing projects. Next, pick an existing project most like your intended solution, and then make a copy of that project. Finally, make modifications to the copy a little bit at a time as you morph the existing project into your solution. After each modification verify that it still runs. If you take a project that runs, make hundreds of changes to it, and then notice that it no longer runs, you will not know which of the many changes caused the failure.