C Compiler

1 Week

Motivation and introduction

Until now, we have used the assembly code applications. Now we will create a compiler for our customized *ASIP Meister* CPUs. With this compiler, we will compile a C-code application and simulate the result in *ModelSim* and *Dlxsim*. This session also introduces about different peripheral where we forward our text/data, and how different libraries are used for their peripherals. The information about creating and using the compiler can be found in Chapter 8 of the Laboratory Script. For every part, that starts like "a)", "b)" … you have to mail the answers and asked files/tables to **sajjad.hussain@kit.edu** and use the topic "asipXX-Session3", with XX replaced by your group number.

Exercises

1) Preparing the Project and Creating the Compiler

1. Based on your ASIPmeister processor design, you can generate GNU Tools accordingly. Which then can be used to compile, assemble and link different assembly and C files. Therefore, you can automatically create a compiler for your individual processor! To get an idea of how retargetable compilers are working, read Chapter 8.1 from the Laboratory Script.
2. Create a project directory as in previous sessions and copy the provided processor *browstd32.pdb* into your project directory, which already includes ADD instructions. [You can also start from the last session's project and just create sub-directories in "Application" directory. This will save time for the following steps 3 &4.]
3. Setup your project directory by adjusting "*env_settings*".
4. Generate VHDL files and GNU tools as in the previous sessions.
5. Please remember that "*AM_tools*" location is kept as in the previous session.

2) Compiling and Simulating the Application

6. Create a subdirectory in your "*Applications*" directory and copy "*/home/asip00/Sessions/Session4/6_for.c*" to this subdirectory. This is the same for-loop example as in the previous session but now in C program. Also, copy the required "*Makefile*".
7. Simulate the program in DLXsim and in ModelSim. Run "*make sim*" and "*make dlxsim*".
   a) How many cycles are required to execute this program DLXsim and ModelSim?
   b) In folder BUILD_SIM, look at the "*6_for.s*" which is generated. Another file "*startup.s*" is used along with the generated "*6_for.s*" to generate TestData.IM/DM files. Just understand and remember the structure of "6_for.s" files if you have to write your own .s file, and how it is being executed along with "*startup.s*".

3) Compiling and Simulating another Application

8. Create another subdirectory in your "Applications" directory and copy "*/home/asip00/Sessions/Session4/app.c*" to this subdirectory. This is a simple example to direct a text to some peripheral devices like LCD or UART. Also, copy the required "Makefile".
9. However, for compiling it, you first need to provide the required libraries to your respective application, i.e. "*lib_lcd_dlxsim.c*", "*lib_uart.c*", "*loadStoreByte.c*", and "*string.c*". Chapter 8.5 describes how to provide these libraries.
10. Simulate the program in DLXsim and in ModelSim. Run "*make sim*" and "*make dlxsim*".

11. After compiling, simulate the application in dlxsim and ModelSim and compare whether the printed results are the same as expected. The dlxsim and ModelSim will print text to a *virtual* LCD/UART. For dlxsim you can forward the LCD/UART output to a file, using the "*-lf*" and "*-uf*" parameters respectively, e.g. "*make dlxsim DLXSIM_PARAM="-da0 –pf1 -lf**lcd.out** -uf**uart.out***" writes output to the file "*lcd.out*" and "*uart.out*". While ModelSim automatically writes to the file "*lcd.out*" and "*uart.out*".
    c) How many cycles are required to execute this program DLXsim and ModelSim?

12. The default GCC compiler optimization is –O0. Try different optimization levels with dlxsim and ModelSim using e.g. "*make dlxsim GCC_PARAM=-O1*" or using "*make sim GCC_PARAM=-O1*".
13. Repeat this benchmarking for all compiler optimization-levels like O0, O1, O2, O3 and O4 for both dlxsim and ModelSim.

    d) Does the application "*app.c*" is executed successfully using different optimization levels? If yes, please fill the following benchmark table:

Optimization Level	**Executed?** [Yes/No]	**Cycle count** ModelSim	**Cycle count** dlxsim
-O0 (default)			
-O1			
-O2			
-O3			
-O4			

Next Session: Adding Custom Instructions
Readings for the next session: Chapters 8.2.3, 3.2.2, ASIPmeister Tutorial