

Preface to Laboratory Exercise 3

Using the Intel FPGA Monitor Program

This document serves as a *Preface* to Laboratory Exercise 3. You are required to read through this document in advance of your Lab 3 practical session. This preface to Lab 3 describes a number of steps that require the use of a physical DE1-SoC board connected to your computer. Assuming that you do not have your own DE1-SoC board (at home), then you will have to perform these steps during your Lab 3 practical session, when we return to campus the week of February 7. However, as mentioned already, read through this document (thoroughly) in advance of your Lab 3 practical session, so that you know what work will be required.

Introduction

For Laboratory Exercises 1 and 2 you used the CPULATOR, which is a great tool for developing assembly language programs that run on the ARM processor, using a simulation of the DE1-SoC board. We will continue to rely on the CPULATOR in lectures, use it for demonstrations of code, and so on. Plus, you will probably use this simulation tool throughout the course as an aid for developing and debugging programs (you should!).

When we return to the University campus starting in the week of February 7, you will have access to a physical DE1-SoC board during your practical sessions, and so you will be able to develop and debug assembly-language programs that run on the actual hardware. To do this, you can't use the CPULATOR, because it does not support the physical DE1-SoC board (it only supports simulation of the board).

For the development of assembly-language programs that run on the actual hardware, you have to use the Intel FPGA Monitor Program application software. This software has many similar features to the CPULATOR, allowing you to examine/set the contents of ARM registers, single-step instructions, set breakpoints, display/set the contents of memory, and so on.

Some of you may have a DE1-SoC board at home (note that this is *not* required), and would like to be able to make use of the ARM processor on your board. If so, then you will need to install the Monitor Program software on your home computer that is attached to your DE1-SoC board.

You will need to learn how to use the Monitor Program for the rest of the ARM-based labs in this course, including Lab 3. The rest of this document provides an introduction to the Monitor Program.

Part I

For this part of the Lab 3 Preface you are to watch a short video that provides an introduction to the *Monitor Program* tool. As you will see in the video, the *Monitor Program* is a full-featured software development environment, much like the *CPULATOR*, that allows you to compile (assemble) and debug software code for the ARM processor. The video also briefly describes the process for installing this software on your own computer, which is relevant only to those of you who have a DE1-SoC board at home. Use your normal *UTOR* credentials to access the video:

<https://web.microsoftstream.com/video/2ac1185c-5a85-4b2d-92c7-63e4aea785d6>

Part II

This part introduces various features of the *Monitor Program*, for developing and debugging code for the ARM processor that is being executed on a *physical* (not simulated) DE1-SoC board. As stated previously, you cannot use the CPULator to control a real hardware board, because the CPULator only *simulates* the features of the DE1-SoC Computer and does support the actual hardware.

A computer running the Monitor Program software must be connected by a USB cable to a DE1-SoC board (the cable has to be plugged into the *USB Blaster* port on the board). The *DE1-SoC Computer* (that includes the ARM processor, memory, and various I/O devices, as you are aware) is implemented as a circuit which is downloaded by the Monitor Program tool into the SoC FPGA device on the DE1-SoC board. You use the Monitor Program to control the ARM processor on the board.

The remaining parts of this Preface to Lab 3 require the use of a DE1-SoC board. Assuming that you do not have a board at home, you will need to perform these steps during your practical session in the University lab room, using your assigned workstation and a DE1-SoC board. But, as stated at the beginning of this document, you are required to read through all of this material (thoroughly) before attending your practical session.

The first step in using the Monitor Program is to set up an ARM software development project. Perform the following:

1. Make sure that the power is turned on for the DE1-SoC board.
2. Open the *Monitor Program* software, which leads to the window in Figure 1.
To develop ARM software code using the Monitor Program it is necessary to create a new project. Select **File > New Project** to reach the window in Figure 2. Give the project a name and indicate the folder for the project; we have chosen the project name *part2* in the folder *tutorial\Part2*, as indicated in the figure. Use the drop-down menu shown in Figure 2 to set the target architecture to the ARM Cortex-A9 processor. Click **Next**, to get the window in Figure 3.
3. Now, you can select your own custom computer system, or a pre-designed (by Intel) system. As shown in Figure 3 select the *DE1-SoC Computer*. Once you have selected the computer system the display in the window will now show where files that implement the chosen system are located. Click **Next**.

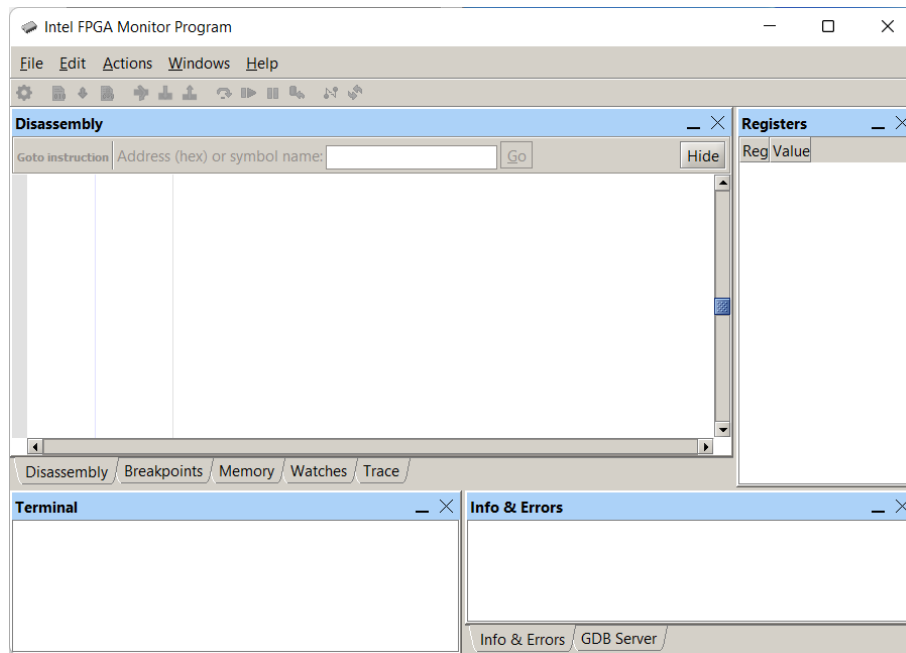


Figure 1: The *Monitor Program* window.

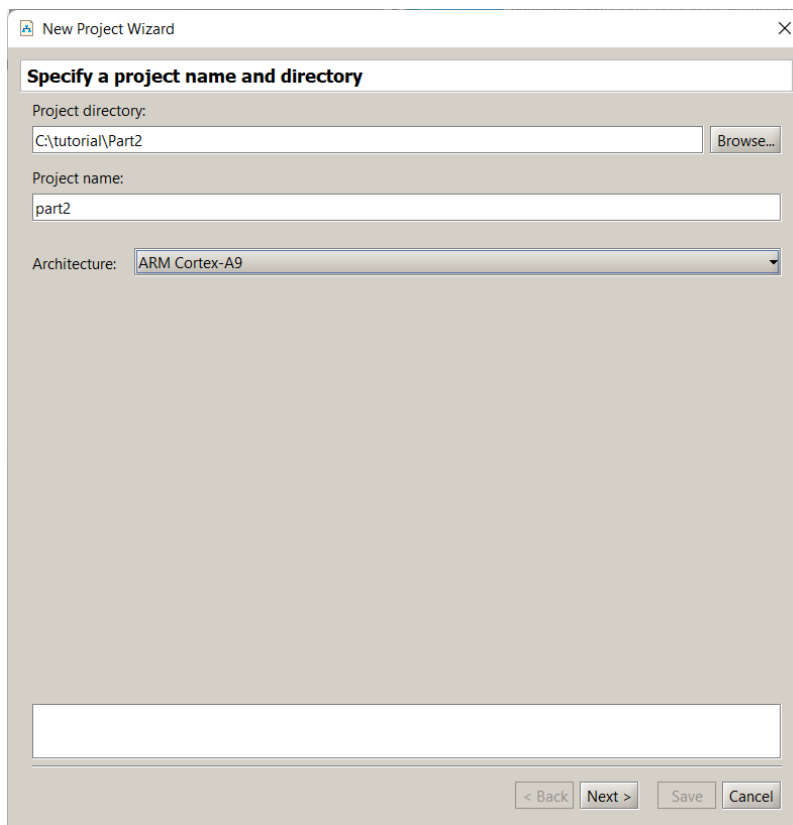


Figure 2: Specify the folder and the name of the project.

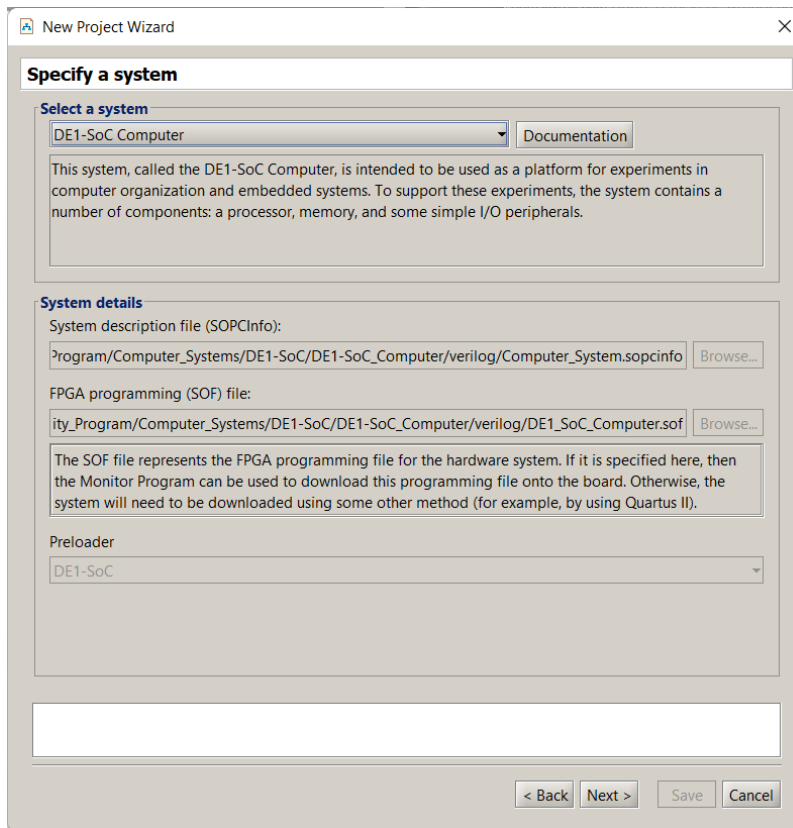


Figure 3: Specification of the system.

4. In the window in Figure 4 you can specify the type of application programs that you wish to run. They can be written in either assembly language or the C programming language. Specify that an assembly language program will be used. The *Monitor Program* package contains several sample programs. Select the box **Include a sample program with the project**. Then, choose the **Getting Started** program, as indicated in the figure, and click **Next**.
5. The window in Figure 5 is used to specify the source file(s) that contain the application program(s). Since we have selected the *Getting Started* program, the window indicates the source code file for this program. This window also allows the user to specify the starting point in the selected application program. The default symbol is `_start`, which is used in the selected sample program. Click **Next**.
6. The window in Figure 6 indicates some system parameters. Note that the figure indicates that the *DE-SoC [USB-1]* cable is selected to provide the connection between the DE-series board and the host computer. This is the name assigned to the Intel USB Blaster connection between the computer and the board. Click **Next**.

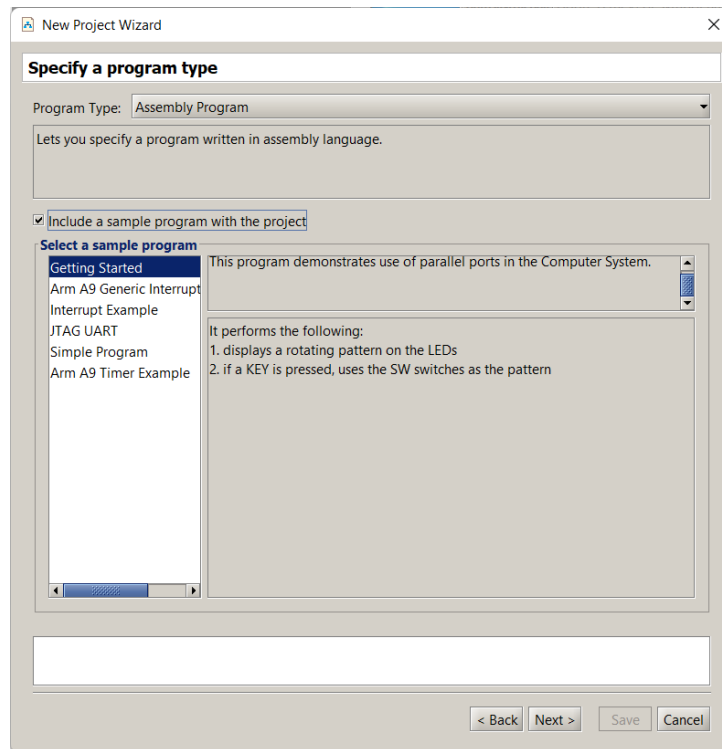


Figure 4: Selection of an application program.

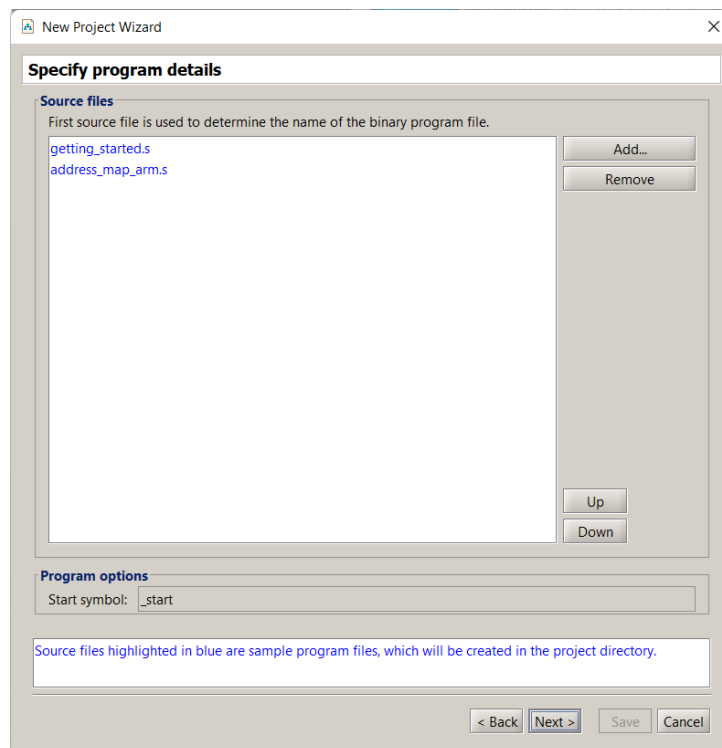


Figure 5: Source files used by the application program.

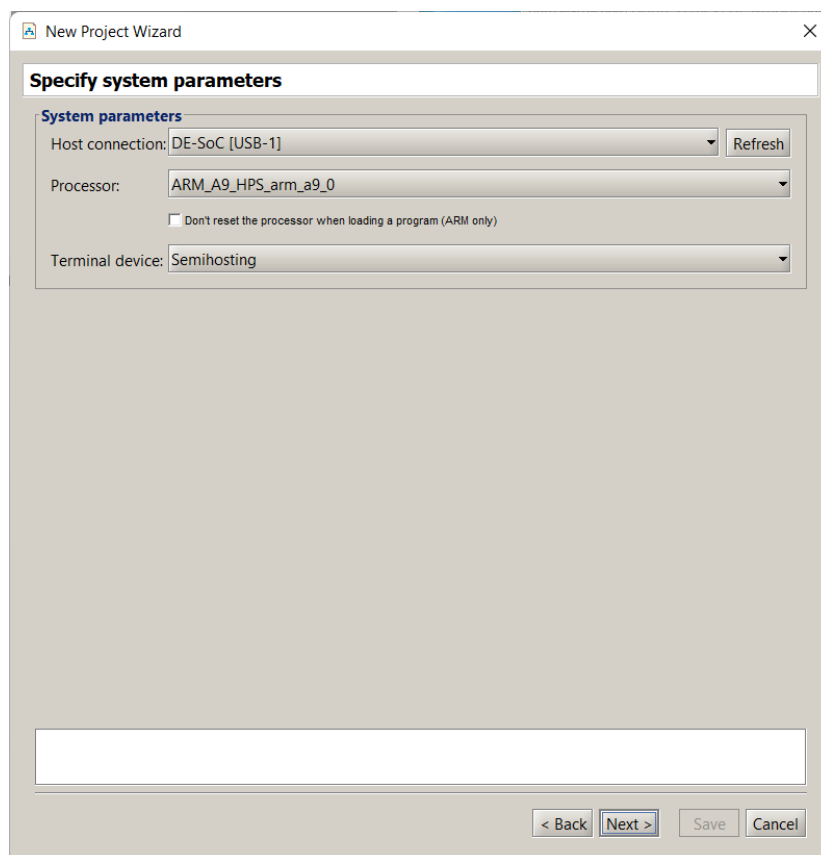


Figure 6: Specify the system parameters.

7. The window in Figure 7 displays the names of Assembly sections that will be used for the program, and allows the user to select a target memory location for each section. In this case only the *.text* section, which corresponds to the program code (and data), will be used. As shown in the figure, the *.text* section is targeted to the DDR3 memory in the DE-series board, starting at address 0. Click **Save** to complete the specification of the new project.
8. Since you specified a new project, a pop-up box will appear asking you if you want to download the system associated with this project onto the DE-series board. Make sure that the power to the board is turned on and click **Yes**. After the download is complete, a pop-up box will appear informing you that the circuit has been successfully downloaded—click **OK**.¹ If the circuit is not successfully downloaded, make sure that the USB connection, through which the USB-Blaster communicates, is established and recognized by the host computer. (If there is a problem, a possible remedy may be to unplug the USB cable and then plug it back in.)

¹The first time that you run the Monitor Program, you may need to respond to a pop-up dialog from MS Windows to allow access to the program (the message may be related to one/both of Java and Quartus).

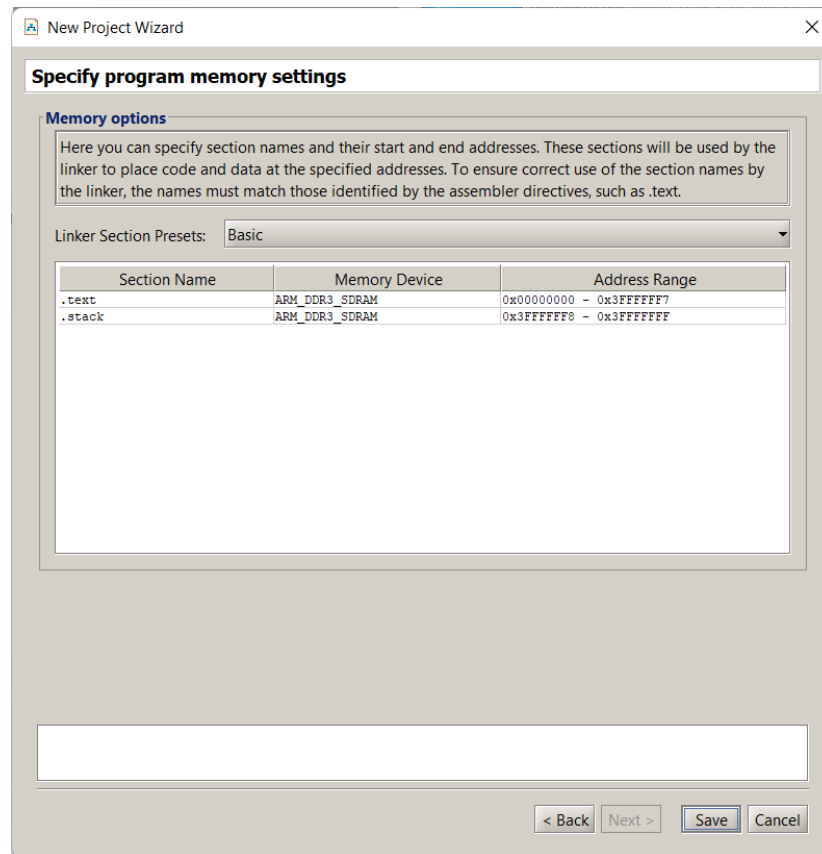





Figure 7: Specify the program memory settings.

9. Having downloaded the computer system into the FPGA SoC chip on your DE1-SoC board, we can now load and run the sample program. In the main Monitor Program window, shown in Figure 8, select **Actions > Compile & Load** to assemble the program and load it into the memory on the board. Figure 8 shows the Monitor Program window after the sample program has been loaded.
10. Run the program by selecting **Actions > Continue** or by clicking on the toolbar icon , and observe the patterns displayed on the LEDs.
11. Pause the execution of the sample program by clicking on the icon , and disconnect from this session by clicking on the icon .

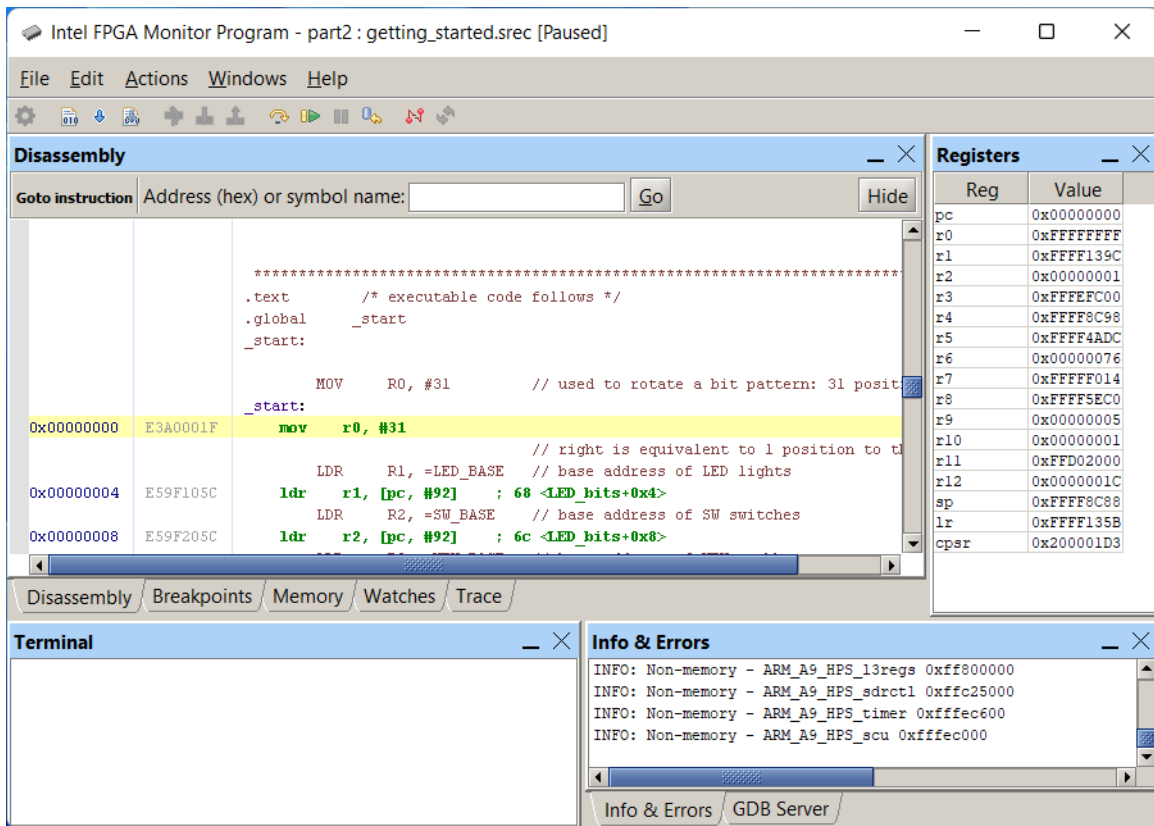


Figure 8: The monitor window showing the loaded sample program.

Part III

Now, we will explore some features of the Monitor Program. You will notice that the discussion below makes use of the ARM assembly-language examples that were used in Laboratory Exercise 1. This is because we are reusing those examples, so that you are already familiar with them and have access to the code. Figure 9 shows the code for a program that finds the largest number in a list of 32-bit integers. This code is reproduced from Figure 1 of Lab 1.

```
/* Program that finds the largest number in a list of integers */

        .text                // executable code follows
        .global _start

_start:
        MOV     R4, #RESULT   // R4 points to result location
        LDR     R2, [R4, #4]  // R2 holds number of elements in the list
        MOV     R3, #NUMBERS  // R3 points to the list of integers
        LDR     R0, [R3]      // R0 holds the largest number so far

LOOP:    SUBS    R2, #1        // decrement the loop counter
        BEQ     DONE         // if result is equal to 0, branch
        ADD     R3, #4        // increment pointer to next integer
        LDR     R1, [R3]      // get the next number
        CMP     R0, R1        // check if larger number found
        BGE     LOOP         // branch if greater or equal
        MOV     R0, R1        // update the largest number
        B       LOOP

DONE:    STR     R0, [R4]      // store largest number into result location

END:     B       END






RESULT:  .word    0
N:       .word    7           // number of entries in the list
NUMBERS: .word    4, 5, 3, 6  // the data
        .word    1, 8, 2

        .end
```

Figure 9: Assembly-language program that finds the largest number.

Perform the following:

1. Create a new folder for this part of the exercise, with a name such as *Part3*. Create a file named *part3.s* and enter the code from Figure 9 into this file. Use the Monitor Program to create a new project in this folder; we have chosen the project name *part3*. When you reach the window in Figure 4 choose **Assembly Program** but do not select a sample program. Click **Next**.
2. Upon reaching the window in Figure 5, you have to specify the source code file for your program. Click **Add** and in the pop-up box that appears indicate the desired file name, *part3.s*. Click **Next** to get to the window in Figure 6. Again click **Next** to get to the window in Figure 7. Notice that the **DDR3_SDRAM** is selected as the memory device. Your program will be loaded starting at address 0 in this memory. Click **Save**. If you are prompted to download the DE1-SoC computer to your board, you can decline, as you already programmed the FPGA chip when performing Part II.
3. Compile and load the program. The Monitor Program will display a disassembled view of the machine code loaded in the memory, as indicated in Figure 10.

4. Execute the program (). When the code is running, you will not be able to see any changes (such as the contents of registers or memory locations) in the display for the Monitor Program, because it does not communicate with the ARM processor while code is being executed. But, if you pause the program then the Monitor Program window will be updated. Pause the program using the icon  and observe that the processor stops within the endless loop **END: B END**. Note that the largest number found in the sample list is 8 as indicated by the contents of register R0. This result is also stored in memory at the label RESULT. The address of the label RESULT for this program is 0x00000038. Use the Monitor Program's Memory tab, as illustrated in Figure 11, to verify that the resulting value 8 is stored in the correct location.
5. You can return control of the program to the start by clicking on the icon  , or by selecting **Actions > Restart**. Do this and then single-step through the program by clicking on the icon  . Watch how the instructions change the data in the processor's registers.
6. Double-click on the pc register in the Monitor Program and then set the program counter to 0. This action has essentially the same effect as clicking on the restart icon  .
7. Now set a breakpoint at address 0x0000002C by clicking on the gray bar to the left of this address, as illustrated in Figure 12. Restart the program and run it again. Observe the contents of register R0 each time the instruction at the breakpoint, which is **B LOOP**, is reached.

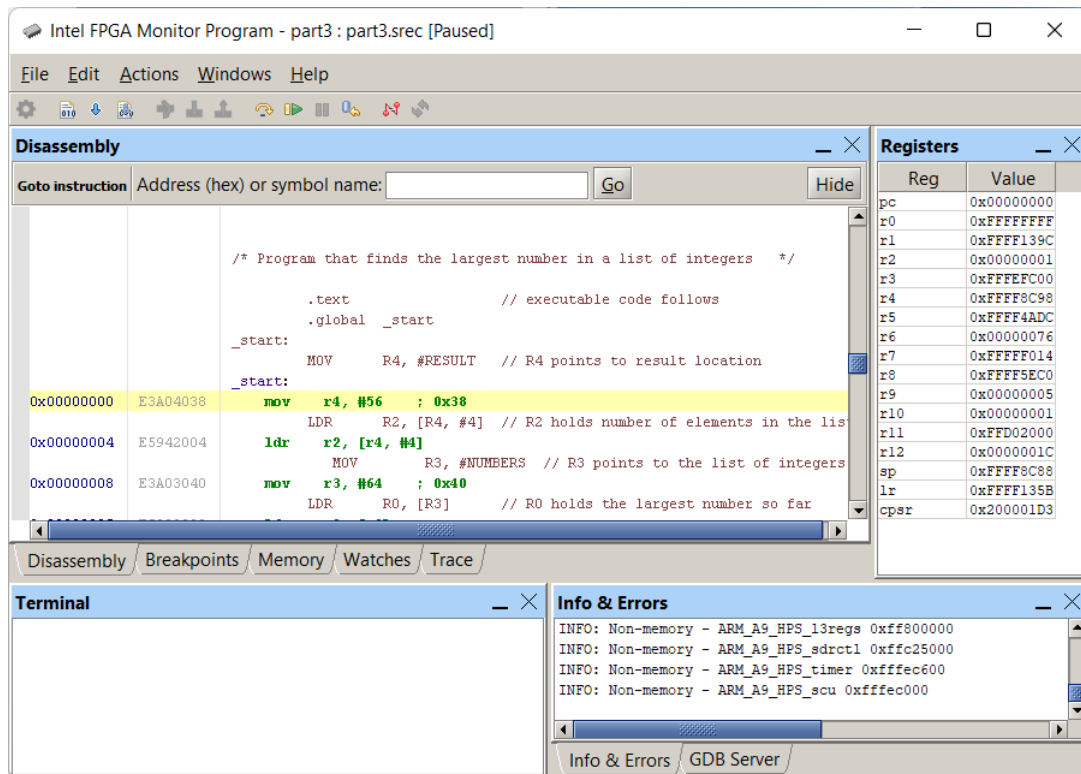


Figure 10: The disassembled view of the program in Figure 9.

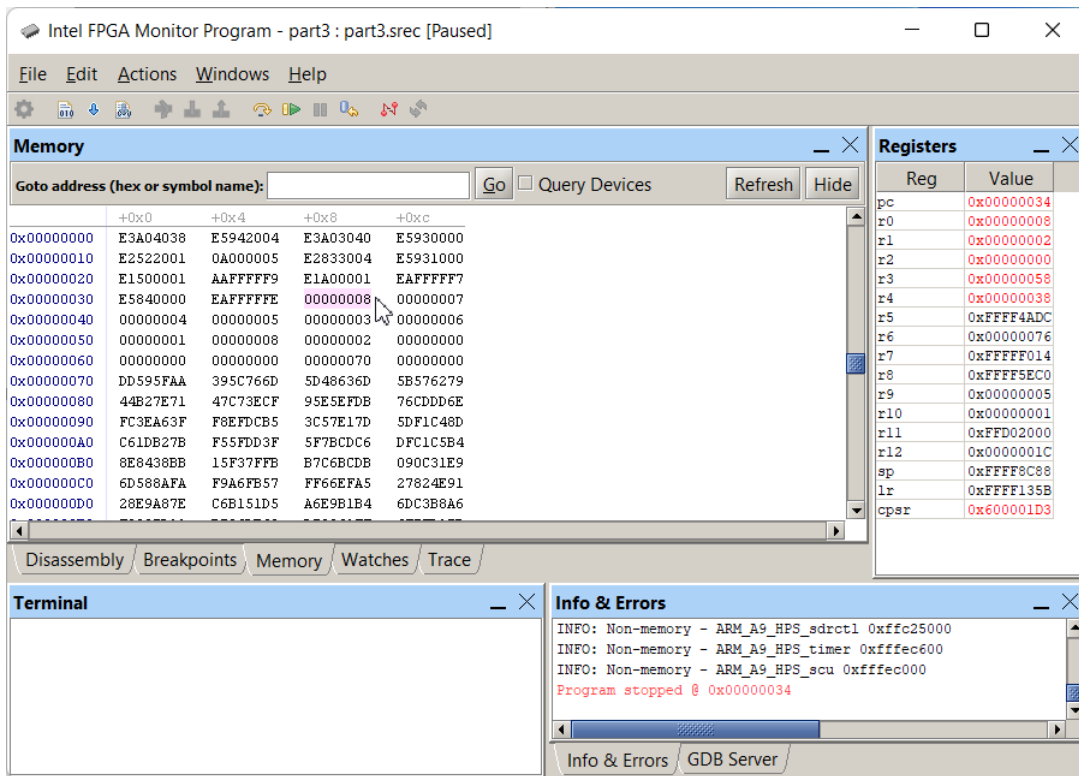


Figure 11: Displaying the result in the memory tab.

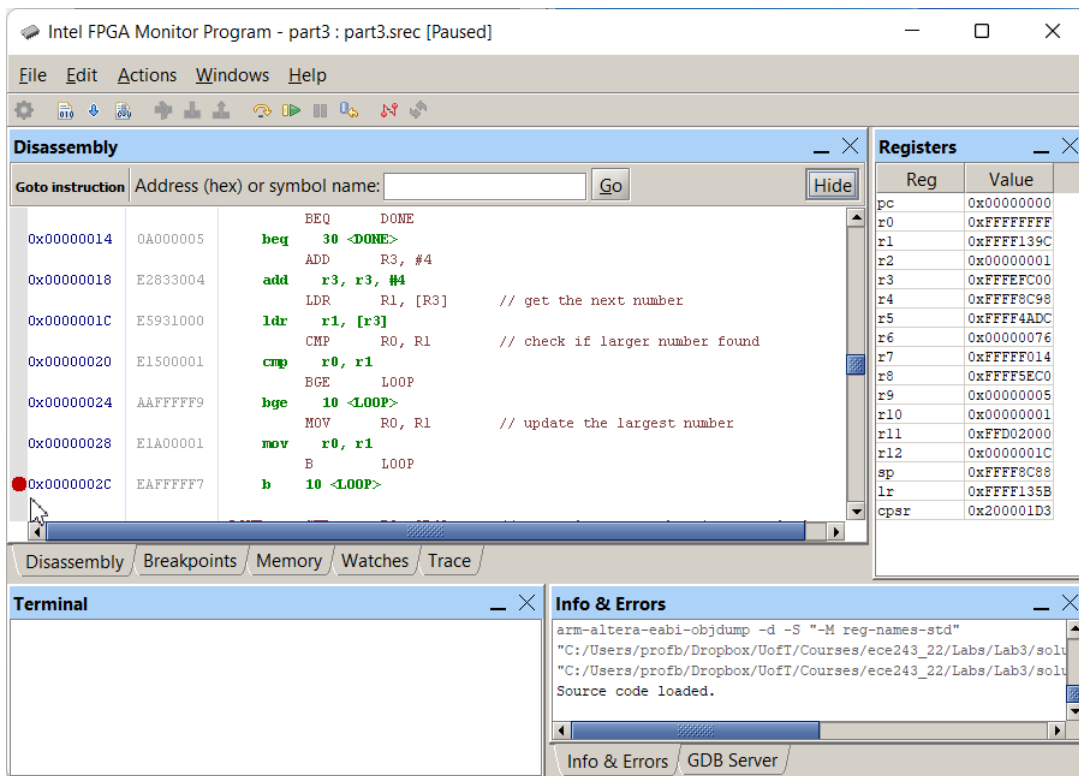


Figure 12: Setting a breakpoint.

Part IV

In this part, you are to run the ARM program that you wrote for Part III of Lab Exercise 1 on the DE1-SoC board, using the Monitor Program. Recall that your program from Part III of Lab Exercise 1 modifies the code from Figure 9 so that it uses a subroutine, called `LARGE`, that finds the largest number in a list.

Create a new folder and a new Monitor Program project to compile and download your program. Run your program to verify its correctness.

Part V

In this part, you are to run the ARM program that you wrote for Part IV of Lab Exercise 1 on the DE1-SoC board, using the Monitor Program. Recall that your program from Part IV of Lab Exercise 1 converts a binary number stored in the memory into four decimal digits, supporting decimal values up to 9999.

Create a new folder and a new Monitor Program project to compile and download your program. Run your program to verify its correctness.

Final Comments

We have provided only a brief introduction to the features of the Monitor Program application software. A more detailed discussion is available in the tutorial *Intel FPGA Monitor Program Tutorial for ARM*, which is provided along with this exercise. This tutorial can also be accessed by selecting **Help > Tutorial** within the Monitor Program.