Lab Experiment 3: Building a Complete Project

1 Introduction

In this lab, you will develop some simple but complete assembly programs and debug with the µVision software. You will be practicing the ARM7TDMI instructions. The objectives are:

- To gain familiarity with a representative subset of the instruction set
- to practice with good programming style
- To learn how to use Simulator to debug code

2 Pre-lab Preparation

Please prepare the lab (e.g., write the needed code, assemble them to eliminate any syntax error, and run your code with µVision simulator) before you come to the laboratory. This will help you to complete the required task on time.

In this lab, we will write code to locate in the on-chip 512K flash region, which begins at address 0x00000000. We will expect our code to run when the reset push-button on the board is pressed. To this end, we will mark our assembly code as the reset handler, and we will declare it globally so that a startup code can find it. Therefore, your project file will consist of two assembly files: one is a startup code and the other your program. You may type in the following program, save it as **mystartup.s**, and include it as one of your project files.

```
                AREA    Reset, CODE, Readonly
                ENTRY   ; The first instruction to execute follows
VECTORS
                LDR     PC,Reset_Addr
                LDR     PC,Undef_Addr
                LDR     PC,SWI_Addr
                LDR     PC,PAbt_Addr
                LDR     PC,DAbt_Addr
                NOP
                LDR     PC,IRQ_Addr
                LDR     PC,FIQ_Addr

Reset_Addr      DCD     Reset_Handler
Undef_Addr      DCD     UndefHandler
SWI_Addr        DCD     SWIHandler
PAbt_Addr       DCD     PAbtHandler
DAbt_Addr       DCD     DAbtHandler
```

```
                DCD     0
IRQ_Addr        DCD     IRQHandler
FIQ_Addr        DCD     FIQHandler

SWIHandler      B       SWIHandler
PAbtHandler     B       PAbtHandler
DAbtHandler     B       DAbtHandler
IRQHandler      B       IRQHandler
FIQHandler      B       FIQHandler
UndefHandler    B       UndefHandler

                IMPORT  Reset_Handler
                END
```

In your program code, you need to declare "Reset_Handler" globally, so that the above startup file can find it. Therefore, the first few lines of your code should be:

```
                GLOBAL      Reset_Handler
                AREA        mycode, CODE, Readonly
                Reset_Handler               ;this label is necessary
```

After the label "Reset_Handler", your assembly code follows. In addition, make the last instruction lines of your code to be

```
stop            B    stop     ;endless loop to make the program hang
                END       ;assembler directive to indicate the end of code
```

Note that addresses ranging from 0 to 0x7FFFF are mapped to the on-chip 512 KB flash memory, which cannot be modified by the storing assembly instructions such as STR. To allocate RAM memory locations to your code or data, you may use, for example:

```
                AREA    Mydata, DATA, Readwrite  ; you don't need it for this lab
```

In addition, as part of lab preparation, you need to learn more about instructions **MOV, LDR, and STR**, and directive **DCD**. You may refer to the Appendix A in the textbook, online at http://www.heyrick.co.uk/assembler/, and http://infocenter.arm.com (searching ARM assembly language).

3. Lab Tasks

3.1 PROGRAM DEBUGGING

Open the project you created for lab experiment 2, and make the executable file ready. Then you may use the debugger for simulation. Follow the steps below.

1. From the **Debug** menu, choose **Start/Stop Debug Session**. This puts you into a debug session and produces new windows, including the Register window and the Memory window. You can open or close any windows using the **View** menu.

2. From the **Disassembly** window, find the machine code corresponding to each line of the assembly program and their addresses in the memory. In your lab report, show how the machine code is stored in the memory with a table of two columns: one column shows the address and the other shows the stored machine code.

3. From the **Debug** menu, select **step** or press F11 to single-step through the code, and record how register values change by observing the **Registers** window, including the R15 (Program Counter).

4. Click on **Start/Stop Debug Session** again to exit from the Debug Session.

3.2 WRITING ASSEMBLY PROGRAMS

For each of the following tasks, you need to

- Create and assemble your code to implement the task **BEFORE** coming to the lab.
- Simulate your code and verify the result with the simulator
- Demonstrate the results to the lab instructor before you leave the lab.

You may consider to create one project file for all the tasks and group your assembly code in different files. Verify the result of your code with simulator, as well as by reading the actual memory contents after execution. Include the detailed procedure in your lab report.

Task 1: Write assembly instructions to,
-store two immediate 32-bit constants, 0XFB,0x33221100 at memory locations 0x40000008 and 0x4000000C. (*try to use EQU too)
-Use assembler directive DCD to allocate memory for 32-bit number, 0x44663377 and so that it will be placed right after your instruction code in the memory. Use ARM instructions to copy thet32-bit numbers into memory locations 0x40000000.
-Verify the result of your code with simulator, as well as by reading the actual memory contents after execution.
Task 2: write assembly instructions to,
- add the first two values (0XFB, 0x33221100)
-store the result in the 0x400000010.
-subtract the second value from third value(0x44663377-0x33221100).
- store the result into memory location 0x40000014.
Task 3: write assembly instructions to,
-rotate to the right 4 times, the first value(0XFB) you stored in the 0x40000008.
-store your result in 0x40000014.
-Verify your result in the registers and check the CPSR.

Task4. write assembly instructions to,
-multiply your first value (0XFB) by 4, and store it in memory location 0x40000018.
-Verify your result in the registers and check the CPSR.

4. Requirements

1. Pre-lab is required. You need to show your assembly code on paper for all the tasks. (with the related flowchart)
2. Lab report is DUE right before the next lab time.
3. You must sign in with your partner at the beginning of the lab, and sign out before you leave the lab.
4. Always disconnect the board from the computer, and clear up the bench top, before you leave. Demonstrate the results to the instructor before you leave. Failure to do so will result in zero point for performance.
5. Save your work in a thumb drive; you may need it for generating your lab report.