Lab 1
Code Size Reduction of Embedded Software
Due: 11/18/2002

1. Objective
This lab is designed to make students familiar with the ARM Software Development Suite, which can facilitate the development of embedded software written in ARM assembly language, C, or C++ programming language. In addition, the lab is designed to introduce the concept of code size optimization to students who are usually trained to not care about the size of a program.

2. Introduction
An embedded system is an electronic system embedded within a given plant or external process to perform a specific function, which has to respond to certain inputs within a prescribed time constraint. The external process comprises both a physical system and also humans performing some supervising or parameter setting tasks. The system can be as simple as a remote controller or as complex as an avionics system. Therefore, some systems are equipped with very simple hardware on which a very simple program is running, whereas other systems are composed of clusters of distributed processors on which many interactive tasks are running in parallel. Because of this, achieving the specific function prescribed by an embedded system, a designer should understand the characteristics and the requirements of the underlying embedded system and make best use of the system resource to achieve the goal. Among the system requirements, code size, power consumption, and performance are of utmost importance. In particular, code size is very important to the applications that have a very limited memory capacity. Thus, in this lab you will be asked to carry out code size optimization for embedded software written in ARM assembly language, C or C++ programming language.

3. Problem Description
You are asked to generate $N$ random integer numbers by using a C function and place these numbers starting at an address defined by yourself. Then, you write a program to sort these numbers in ascending order. You can use any sorting algorithm that
would result in smaller code size and execution time. Print out the total number of
random numbers, the number sequence before and after sorting. Solve this problem
using the following methods, respectively:
(A). Write a C++ program to carry out this job. You have to define the sorting
algorithm as a class function. This program is called \( PA \).
(B). Write a C program to carry out this job. This program is called \( PB \).
(C). Write a C program which mixes some in-line assembly code that carries out the
sorting function. This program is called \( PC \).

Compare the code sizes and execution time (or the number of CPU cycles) obtained
by the above three programs. When you use an ARM compiler or the ARM assembler,
you can invoke any compiling or assembling options to minimize code size, execution
time, or both. So you have to report the results that fill up the entries in Table 1 and
report the compiling or assembling options used for each individual case in Table 2.
Note that if you use the CodeWarrior to develop your software, you might have to
find out for the options used by the CodeWarrior the corresponding options used by
the Command-line commands. The execution time of a program (i.e., the number of
clock cycles spent for executing a program) must be obtained form running the
program on ARMulator.

Table 1. Results for experimenting with the four programs (\( N=500 \))

<table>
<thead>
<tr>
<th></th>
<th>( PA )</th>
<th>( PB )</th>
<th>( PC )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized for</td>
<td>Code size (bytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>code size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimized for</td>
<td>Code size (bytes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>execution time</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Execution time (cycles)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. Compiling or assembling options used for each case (\( N=500 \))

<table>
<thead>
<tr>
<th></th>
<th>( PA )</th>
<th>( PB )</th>
<th>( PC )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimized for</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>code size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Optimized for</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>execution time</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
4. Evaluation

You have to hand in a report that summarizes your results and point out the specific features of your software implementation. If you have used a more sophisticated sorting algorithm, clearly indicate this point. The name of reports is designated as report_your_student_ID.doc, where your_student_ID represents your student ID number. Rename each of the three programs as PX_your_student_ID.XXX, where PX is any of PA, PB, and PC; XXX indicates the file extension. You have to ftp both the three programs and the report to the directory called LAB1 in our web-site for this course, respectively. You also have to hand in a copy of the report by the due date. Your score will computed based on the following criteria:

(A). Correctness of your program.

(B). The code size and the execution time presented in Table 1.

(C). The comments you add in the programs.

(E). Others.

Do this lab by yourself. If you are caught because of plagiarism, you will get zero points for this course and be punished accordingly.

5. Bill of Materials

You are asked to learn the basics of using ARM Developer Suite (ADS) in this lab. So you probably have to study the following manuals:


This manual introduces to you the basic components in ADS. This includes command-line development tools, GUI developments (Debugger and Project management tools), some utilities, and flash downloader.


This manual teaches you how to interwork ARM and Thumb programs and Mix C, C++ and assembly codes in the same program. Some other materials in this manual are also very useful later. This includes the ARM-Thumb Procedure Call Standard, Handling Processor Exceptions, and Writing Code for ROM.


This manual teaches you the basics of ARMulator, an instruction accurate simulator for your program development. Another important subject covered by this manual is the concept of Semihosting. Semihosting provides code running on an ARM target use of facilities on a host computer that is running an ARM debugger. Examples of such facilities include the keyboard input, screen output, and disk I/O. This is introduced in Chapter 5.

Introduce to you the details of using ARM GUI platform to develop your software.

Introduce to you the various kinds of Debuggers, especially the one that you will use in your software development. This is AXD.

Introduce to you the ARM and Thumb assembler language, especially the directives that direct an assembling task. This manual also introduces to you the details of ARM and Thumb instruction sets.


All these manuals are available to you. We will put a copy of these manuals on the course web site for your convenience.