COE 306, Term 171

Introduction to Embedded Systems

Assignment# 1
Due date: Sunday, Oct. 8, 2017

Q.1. Look for a consumer product, e.g. a toy or a small home appliance that incorporates an embedded system. It will be good if you can find a locally available product, and include pictures showing its computer components.

Note: You may need to disassemble the product to be able to verify that it does incorporate an embedded system. If this is the case, please exercise caution and follow safety procedures to avoid any injuries. If unsure about the safety of an action, consult lecture or lab instructors.

(a) What feature of the product suggested to you that it may incorporate an embedded system?

(b) What is the model number of the microcontroller chip used in this product?

(c) What is the architecture of this microcontroller?

(d) What are the main specifications of this microcontroller? On-chip memory, integrated peripheral devices, maximum frequency, power rating (volts, amperes, watts), any other notable specifications.

(e) Provide the URL of the microcontroller datasheet.

(f) Include a picture of the product in its original condition, and another picture of its main board, showing the main microcontroller.

Q.2. Consider the C code given below:

```
volatile static int sum = 0; // assume sum is a memory variable
for (int i=0; i<8; i++){
      if (i<4)
            sum = sum + 2*i;
      else
            sum = sum + 3*i;
}
```

(a) Write the given C code in ARM assembly without the use of conditional instructions except branch. Simulate your program using VisUAL ARM emulator and include a snapshot to show that your program produced the correct result of sum.

(b) Write the given C code in ARM assembly with the use of conditional instructions. Simulate your program using VisUAL ARM emulator and include a snapshot to show that your program produced the correct result of sum.

(c) Compare your assembly codes in part (a) and part (b) in terms of number of instructions in each code and the number of instructions executed in each code.

(d) Compile your C program to ARM assembly code and compare your handwritten assembly program in part (b) with the compiler-generated assembly code using LPCXpresso IDE. Explain any differences between the two assembly programs.

Q.3. Consider the C code given below:

```
volatile static int x; // assume x is a memory variable
x = x * 217;
```

(a) Write the given C code in ARM assembly without the use of multiplication instructions. Simulate your program using VisUAL ARM emulator and include a snapshot to show that your program produced the correct result of sum.

(b) Compile your C program to ARM assembly code and compare your handwritten assembly program in part (b) with the compiler-generated assembly code using LPCXpresso IDE. Explain any differences between the two assembly programs.

How to compile your C program into assembly

To compile your C program into ARM assembly, use the LPCXpresso IDE (used in the lab) to create a project, and write your C code in the main function. Then, perform the following steps to instruct the compiler to generate the assembly code:

1. Go to *Project Properties*.
2. Select *C/C++ Build*, then *Settings* from the left-hand side menu.
3. In the main pane, select *MCU C Compiler*, then *Miscellaneous*.
4. Add the following into the *Other flags* field:

   -Wa,-ahlnds=${OutputFileBaseName}.asm -O1

5. In the main pane, select *MCU C Compiler*, then *Architecture*. Set the Architecture to ARM7TDMI and uncheck Thumb mode.

Now, when you compile (build) your project, you will get a file under the Debug directory with extension .asm containing the assembly code generated from your C program using ARM instruction set.

You can also generate the ARM assembly code for a C program, e.g. program.c, using the command line, without using the LPCXpresso IDE, or any other IDE, using the GCC ARM cross-compiler package: gcc-arm-none-eabi. The official download page for the gcc-arm-none-eabi package is: https://launchpad.net/gcc-arm-embedded/+download. You can use the following command: **arm-none-eabi-gcc -S program.c**