Project: C to Assembly Language

COSC 216.001

Due Date: October 25, 2021, 11:59pm

1 Overview

Each student is asked to convert a number of C-subroutines into VisUAL compatible ARM Assembly Language.

1.1 Objectives

- Understanding Assembly language code
- Explore the difference between higher-leVel language and Assembly

2 Detailed Description

2.1 Introduction

The ARM architecture has quickly become the go-to platform for mobile and embedded systems. ARM supports a software toolchain built around the Unified Assembler Language (UAL). In this project you will use a subset of the ARM assembly code that is compatible with the VisUAL ARM emulator system. Your task is to write Assembly language programs which correspond to provided C programs. In Blackboard, you will find three sets of C code, along with the corresponding header files and a makefile for each one. The C code descriptions are included below.

2.2 What to do

You are provided with three applications written in C. The three programs are described in the subsections below. You need to use the subset of Assembly Language instructions available in the VisUAL emulator to re-implement these programs.

2.2.1 change.c

Computes the number of dollars, quarters, nickels, dimes, and pennies are required to provide correct change:

The amount of change is set in main. A series of computations are performed to determine the number of dollars, quarters, dimes, nickels, and pennies.

Your assembly language version should start with the amount of change in R3, with dollars, quarters, dimes, nickels, and pennies stored in 4 byte values in: FP-4, FP-8, FP-12, FP-16, and FP-20; respectively.

2.2.2 concat.c

Concatenates 2 strings. The key things is the subroutine allocate(), that allocates memory from the heap. You should create this is an independent function, and call that function from your main routine.

The amount of change is set in main. A series of computations are performed to determine the number of dollars, quarters, dimes, nickels, and pennies.

Your assembly language version should start with two strings in R3 and R1 with their lengths in R2 and R0, respectively. You should allocate enough room from both strings and the subsequent null from the heap. Your should then copy each string into the newly allocated memory and return the address in R0.

2.2.3 hamming.c

Uses hamming code parity to correct errors in a 16-bit value.

The input value for correction should be assumed to be in R3, and the corrected 16-bit value should be output to R0. The algorithm itself looks a bit complex but isn't really that tough to code. The issue will likely be translating for-loops into assembly language.

2.3 Running your code

Please use the VisUAL ARM emulator to test your code. You can download VisUAL at: `https://salmanarif.bitbucket.io/visual/downloads.html`

3 Submission

Each of your source code files must have a comment near the top that contains your name, StudentID, and M-number.

The project should be submitted via Blackboard by **October 25, 2021, 11:59pm**. Follow these instructions to turn in your project.

You should submit the following files:

- change.s
- concat.s
- hamming.s

- *any other source files your project needs*

The following submission directions use the instructions that we will use for all projects this semester.

1. create a directory for your project:

```
mkdir c2asm
```

2. create (or copy) all of your source files in this directory. Example: To copy a file called `example.s` into your `c2asm` directory:

```
cp example.s c2asm
```

3. change parent directory of your project directory:

```
cd c2asm/..
```

4. Create a tar file named `<user_name>.tar.gz`, where `<user_name>` is your studentID and `<proj_dir>` is the directory containing the code for your project, by typing:

```
tar -czf <user_name>.tar c2asm
```

5. Finally, submit the compressed tar file to Blackboard in accordance with the class policies.

Late assignments will not be given credit.

4 Grading

While this rubric is subject to change based on class performance, the current grading rubric for this assignment is as follows:

component	value
Correctness	40
Completeness	40
Code Quality	20