CSCI 2500 — Computer Organization
Homework 2 (document version 1.3)
Performance Calculations and Compiling into MIPS

Overview

- This homework is due by 11:59:59 PM on Monday, October 7, 2019.
- This homework is to be completed **individually**. Do not share your work or code with anyone else.
- You **must** use C for the programming portion of this homework assignment, and your code **must** successfully compile via `gcc` with absolutely no warning messages when the `-Wall` (i.e., warn all) compiler option is used. We will also use `-Werror`, which will treat all warnings as critical errors.

Homework Specifications

In this second homework assignment, you will work on a mixture of textbook problems and C code.

First, start with some "warm-up" exercises, which you will **not** submit as part of this assignment. In other words, do these "warm-up" exercises as practice and to prepare to work on the actual problems you will submit for credit.

Warm-Up Exercises (for Practice)

1. **Textbook Problem 1.3**
2. **Textbook Problem 1.5**
3. **Textbook Problem 1.7**
4. **Textbook Problem 1.13 (all sub-parts)**

Homework Problems (to Submit for Credit)

Use whatever software you like to write your answers to the textbook problems below. You must produce a PDF to submit for this assignment. Please name your PDF `hw2.pdf`. These will be manually graded by our TAs.

1. **Textbook Problem 1.6**
2. **Textbook Problem 1.9 (all sub-parts)**
3. **Textbook Problem 1.12 (1.12.1 and 1.12.2 only)**
4. **Textbook Problem 1.14 (all sub-parts)**

Coding Problem (to Submit for Credit)

Write a C program to take as input (via `stdin`) a valid assignment statement in C and generate MIPS assembly code to perform the given calculation(s). You can assume that each C variable name is one lowercase letter (e.g., `a`, `b`, `c`, etc.) and of type `int`. Further, positive `int` constants are also allowed as part of the given expression. For this assignment, you only need to support the addition (`+`) and subtraction (`-`) operators.

Note that you should use `isspace()`, `islower()`, `isdigit()`, `scanf()`, etc. to parse the input.

The MIPS code you generate must make use of registers `$s0,$s1,...,$s7` to correspond to C variables and registers `$t0,$t1,...$t9` to correspond to any temporary variables you need. Variables in MIPS should match those in C from left to right, meaning that the final result of the assignment statement must end up in register `$s0`.

You can assume that you will not need more than the specific MIPS registers listed here.

Make sure to use the proper I-format instruction when the operand is a constant. If the constant happens to be the first operand of the very first addition or subtraction, use addition with a zero register and this constant, as shown in one of the examples.

Below are a few example runs of your program. In the first example, register `$s0` corresponds to C variable `f`, `$s1` corresponds to `g`, and `$s2` corresponds to `h`.

```
bash$ ./a.out
Please enter a valid C assignment statement:
f = g + h - 42;
The MIPS pseudocode is:
add  $t0,$s1,$s2
addi $s0,$t0,-42
bash$ ./a.out
Please enter a valid C assignment statement:
x = q - 12 + j;
The MIPS pseudocode is:
addi $t0,$s1,-12
add  $s0,$t0,$s2
bash$ ./a.out
Please enter a valid C assignment statement:
t = 9 + s - 1;
The MIPS pseudocode is:
addi $t0,$zero,9
add  $t1,$t0,$s1
addi $s0,$t1,-1
```

```
bash$ ./a.out
Please enter a valid C assignment statement:
a = x - y + 13 + x - a;
The MIPS pseudocode is:
sub  $t0,$s1,$s2
addi $t1,$t0,13
add  $t2,$t1,$s1
sub  $s0,$t2,$s0
bash$ ./a.out
Please enter a valid C assignment statement:
Woooo woooo woooo;
The MIPS pseudocode is:
Parsing error...
```

Be sure to match the above output formatting and ordering of operands exactly as shown (to ensure full credit on Submitty).

Note that if an invalid input occurs, output `fprintf( stderr, "Parsing error...\n" )` and return with `EXIT_FAILURE`.

Submission Instructions

For this assignment, you will submit both your code and your PDF (i.e., `hw2.pdf`) with your answers to the textbook problems to submit.

Before you submit your code, be sure that you have clearly commented your code (this should not be an after-thought). Further, your code should have a clear and logical organization. In general, each function should easily fit on a single screen and have a clear (and clearly documented) purpose. Variable names and function names should be intuitive and meaningful. And use a consistent logical method for indentation.

To submit your assignment (and also perform final testing of your code), please use Submitty.

Note that the test cases for this assignment will be available on Submitty a few days before the due date and will include hidden test cases.

Also as a reminder, your code **must** successfully compile and run on Submitty, which uses Ubuntu v18.04.3 LTS.