CS-382 Computer Architecture and Organization Fall 2022

Lab 5 · Conditionals

Lecturer: Shudong Hao *Date: See Canvas*

1 Task 1: Simple Conditionals

The first task of this lab is to write a program that check that if the given three points (x, y) can be a right triangle. A right triangle is the one that has an angle of 90 degrees, and can be verified by the following formula:

$$a^2 = b^2 + c^2 \tag{1}$$

where a, b, and c are the lengths of three sides of the triangle. Given two points (x_1, y_1) and (x_2, y_2), the distance between them can be calculated by

$$d_{1,2} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}. \tag{2}$$

Since we haven't learned about calculating square roots and floating points, we can simply take the squared value, *i.e.*,

$$d_{1,2}^2 = (x_1 - x_2)^2 + (y_1 - y_2)^2. \tag{3}$$

The data segment is declared as follows:

```
.data
p1:  .quad    0,  0
p2:  .quad    0,  2
p3:  .quad    2,  0
yes: .string  "It is a right triangle."
no:  .string  "It is not a right triangle."
```

where the two numbers in each point represent (x, y) coordinates, respectively.

If it is a right triangle, you'd need to load the **address** of string `yes` to register `X1`; otherwise `no` to register `X1`. Once you have loaded the registers, you can safely exit your program. Later after we have learned system calls, we will revisit this lab, and see how we can print the string out (not using `printf()`).

2 Task 2: Debugging Assembly Using `gdb`

To check if our programs are correct, we would have to rely on `gdb` again (sorry, still not `printf()` yet!). A very comprehensive tutorial of using `gdb` to debug assembly program is in Appendix B.3 of the textbook. Read through the section before you start this task.

In this task, you'd need to write a report on using `gdb` to debug two programs:

(1) The dot product program in the previous lab:

- You need to provide sufficient screenshots of `gdb` to show that your program is correct or incorrect. If you program is correct, use `gdb` to prove it; if not, state which part of your code is wrong and modify the code, and show the corrected result. All of these need screenshots from `gdb`;
- Do not change your code from previous lab before debugging;

(2) The triangle checking program in this lab:

- First, use the data segment provided above to show that you have successfully judged the triangle type;
- Then change the three points in the data segment to show that you can also detect that it's not a right triangle;
- All of these need to have explanation and sufficient `gdb` screenshots as proof.

3 Requirements

- **Note** your code is a **complete** assembly program (not just a sequence of instructions). It should be able to assemble, link, and execute without error and warnings. When executed, the program should finish without problems (also **without any outputs**);
- If your code cannot assemble, you get no credit – this is the same as C programs that cannot be compiled;
- For task 1, the three points in the data segment in your submitted code should not be the same as the example provided above;
- `MUL` instruction can be used for multiplications;
- Avoid using registers `X29` and `X30`;
- You have to put comments on each line. Without comments your code will be penalized heavily;
- Put your name and honor code pledge at the top of your code in comments.

Deliverable

(1) Assembly code for task 1;
(2) If your code from previous lab is incorrect, submit a modified code for the previous lab as well;
(3) A **PDF** lab report: see task 2. Any report in non-PDF format is not allowed. Note for the screenshots do not take pictures.

No need to zip all files; just submit all files separately on Canvas.