Session 2 Introduction To RISC-V Assembly

1.1 Exercises

Exercise 1 Write a RISC-V program that calculates the following: $c = a^2 + b^2$. Use the data section to reserve memory for a, b, and c.

Exercise 2 Given the following code:

```
switch (k) {
        case 0: f = i+j; break;
        case 1: f = g+h; break;
        case 2: f = g-h; break;
        case 3: f = i-j; break;
}
```

Assume that f, g, h, i, j, k are in registers *s0* to *s5* respectively. Convert this code to RISC-V assembly assuming the switch is converted to successive if-then-else statements.

Exercise 3 In exercise 1 of the first exercise session we wrote a C program to iteratively compute the factorial of a given integer. Translate this program to RISC-V. The integer can be provided in the data section.

Exercise 4 Write a RISC-V program that calculates: $c = a^b$. First assume $b >= 2$. What changes when $b >= 0$?

Exercise 5 Write a RISC-V program that multiplies all numbers in an array with a constant number without using the mul instruction.

Exercise 6 In exercise 6 of the first exercise session we wrote a C program to find the length of a string without using *strlen*. Translate this C program to RISC-V. The string can be provided in the data section. The resulting length can be stored in register *a0*.

Exercise 7 Write a program that searches for a given zero-terminated substring in a string and returns 1 if it is present, 0 if it isn't. Define the strings in the data section and place the result in register *a0*. First write a solution assuming that the characters of the string are 32-bit words (use *.word* instead of *.string*). What changes if the characters are bytes (using *.string*)? In what way does this affect performance?