Department of Electrical & Computer Engineering
University of California, Davis

EEC 170 – Computer Architecture
Spring Quarter 2025

Laboratory Exercise 1: Learning RISC-V Assembly Language

Due Date: [Friday April 18, 11:59pm]
Full Points 150

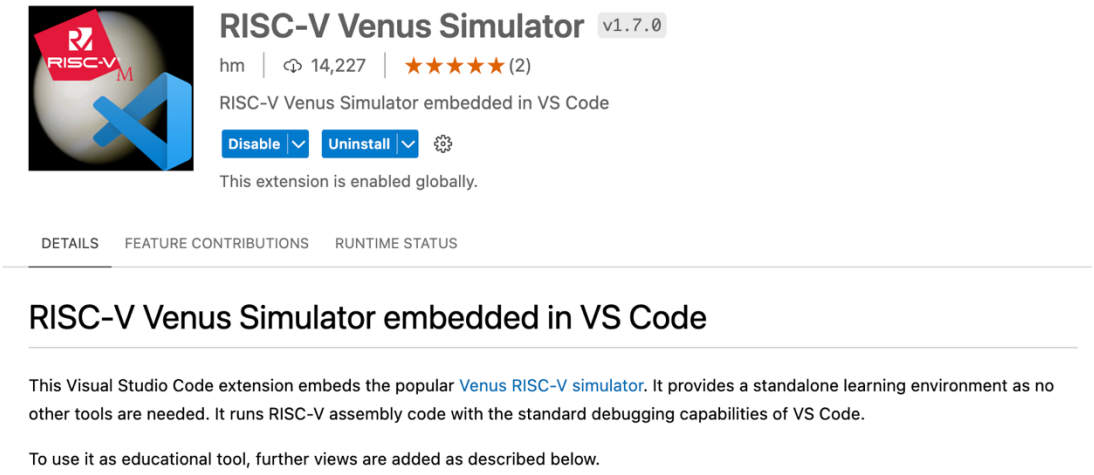
Objectives of Lab 1

1. Set up the Venus RISC-V simulator
2. Learn some basic RISC-V assembly language instructions
3. Learn how to print to the terminal with system calls

Step 1 - Download and install the Venus RISC-V Simulator [25 points]

1. Download Microsoft Visual Studio Code from the link below: https://code.visualstudio.com/

2. After installation, you will need to install the Venus RISC-V extension from the extensions marketplace on Venus code, which looks like this.



Step 2 - Run your first program

1. Open lab1_skel.s program (that comes with this lab assignment) in VSCode.

2. Read the statements that begin with #, which denote comments. Note that an assembly language program has two sections

.data where you declare all your variables.
.text where you put your program in the form of RISC-V instructions, one per line

3. Printing is done by passing the pointer to the string or the value to be printed in register a1 or x11 and a code that tells the system what to print (integer, string, character, etc). in register a0 or x10. This is followed by the command **ecall.**

4. See https://github.com/ThaumicMekanism/venus/wiki/Environmental-Calls for more documentation system calls and other aspects of the Venus simulator.

5. Run the program in the debug mode. You can single step through each instruction.

6. The program shows you several examples of printing strings and integers and how to do a simple computation.

Step 3 - **Your first RISC-V Assembly Language Program** [25 points]

a) Modify the skeleton code provided to *subtract* Y from X. You may simply directly modify the appropriate instructions in part 1, or you may copy-paste them below and modify.
b) Modify (or add) instructions to print "X – Y = {ANS}" where ANS is the actual value of subtracting Y from X.

Step 4 – **Translate a C function into RISC-V Assembly** [100 points]

The next portion of the lab will give you practice with:

- Writing loops and conditional branches in assembly
- Loading data from memory
- Translating C code into assembly (being a human compiler)

On the last page of the lab manual is a C program that contains two functions: main, and a function countOccurrences which is called by main and counts the number of times a particular character occurs in a string. Your task is to translate this C program into assembly. Here are some suggestions/hints:

- Refer to example 2.9.2 in the textbook for a worked example of compiling a very similar program to copy strings.
- To simplify things a bit, we will be working simply with a hardcoded string (declared in the .data section of your program). To gain practice with procedure calls, you are encouraged to structure your implementation of "countOccurences" as a procedure and "call" it via jal. However, this is not required and you may also simply "inline" your function directly into your main assembly program. In fact, compilers will sometimes inline functions as an

optimization to avoid the overhead of a function/procedure call (see: https://en.wikipedia.org/wiki/Inline_expansion)

- Try your code with different strings and characters to make sure you get the expected result. However, when submitting your code, please use the original string/char provided in the template. I.e. count the number of occurrences of “a” in the string:

```
The goal of this lab is to learn RISC-V Assembly Language
```

- Recall that characters are represented using 8 bit (1-byte) ASCII codes – see section 2.9 in the textbook for more detail.

What to submit?

The modified lab1.s file and screenshots from the RISC-V simulator of your results for both part1 and par2 of the exercise.

```c
#include <stdio.h>

int countOccurrences(char *str, char ch) {
    int count = 0;

    // Loop through the string
    for (int i = 0; str[i] != '\0'; i++) {
        if (str[i] == ch) {
            count++;
        }
    }

    return count;
}

int main() {
    char str[100], ch;
    int count;

    // Input string from user
    printf("Enter a string: ");
    fgets(str, sizeof(str), stdin);  // Read string

    // Input character to search from user
    printf("Enter a character to find its occurrences: ");
    scanf("%c", &ch);

    // Count occurrences of the character
    count = countOccurrences(str, ch);

    // Print result
    printf("The character '%c' occurs %d times in the string.\n", ch, count);
    return 0;
}
```