**Department of Electrical & Computer Engineering**

**University of California, Davis**

**EEC 170 – Computer Architecture**

**Fall Quarter 2024**

**Laboratory Exercise 1: Learning RISC-V Assembly Language**

*Due Date: [Friday October 10]*

*Full Points 150*

**Objectives of Lab 1**

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

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

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

* Open lab1\_skel.s program (that comes with this lab assignment) in VSCode.
* Read the statements that begin with #, which denote comments. Note that an assembly language program has two sections

**.data** where you declares all your variables.

**.text** where you put your program in the form of RISC-V instructions, one per line

* 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.**
* See <https://github.com/ThaumicMekanism/venus/wiki/Environmental-Calls> for more documentation system calls and other aspects of the Venus simulator.
* Run the program in the debug mode. You can single step through each instruction.
* 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]

* Modify the program to compute the subtract 2 numbers.
* Modify the message so that it prints X-Y = whatever the answer is

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

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;

}

// Program to test the function countOccurrences.

#include <stdio.h>

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;

}

* Follow the instructions in the lab skeleton file to translate the function countOccurences shown above.
* See the example on Page 116 of your textbook. The function you will write is very similar.
* The RISC-V assembly for data declarations and testing the program are given to you in the skeleton file
* We are going to hardcode the string and the character in the data declarations instead of getting it from the user
* Try your code with different strings and characters to make sure you get the expected result

**What to submit?**

The attached report sheet with screenshots and code provided. Please limited your inserted code and screenshots to the specified areas (screenshot your code and shrink the images when needed)

**Report Sheet**

**Step 2 - Run your first program**

|  |
| --- |
|  |

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

* Modify the program to compute the subtract 2 numbers.

|  |
| --- |
|  |

* Modify the message so that it prints X-Y = whatever the answer is

|  |
| --- |
| **Created Message 7 to print X-Y =** |

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

|  |
| --- |
| **######## PART 2 BEGIN #############**  **la x5, str # x5 = &str[0]**  **li x6, 0 # x6 =0 or count initialized to 0**  **la x7, ch # x7 = &ch address of ch is in register x7**  **lbu x7, 0(x7) # x7 = ch the character ch is in register x7**  **li x8, 0 # x8 = 0 x8 is the index variable of the for loop i, which is set to 0**  **#your code goes here. Call your function countOccurences**  **countOccurences:**  **lb x9, 0(x5) # x9 = str[0]**  **beq x9, zero, printCount # if str[0] = /0 end loop**  **beq x9, x7, matchincrement # if str[0] == ch counter ++ else continue**  **addi x5, x5, 1**  **j countOccurences # jump back to start**  **matchincrement:**  **addi x6, x6, 1**  **addi x5, x5, 1**  **j countOccurences**  **# Print routine is given for you. So, don't have to modify anything, below this**  **printCount:**  **la a1, msg4**  **li a0, 4**  **ecall** |