CS204 - Computer Architecture
Practice Lab - 4 and 5
Date: 10th Feb 2025.
Not graded. Deadline: 3rd Mar 2025. 12Noon. Attempt at the earliest.
Working with RISC - V using Venus

This is our fourth lab with RISC-V ISA and *Venus* simulator. Below tasks are not exhaustive but are only indicative for your practise. Keep up the practice of arranging your code using the assembler directives. Also, continue to preload the data you want to work with.

As you start exploring with control instructions, having a visual of PC of each instruction will be very handy. So, start observing PC in *Venus* https://venus.cs61c.org/ (Dont worry about Cache right now, we'll look at it in-depth in Module 3 of the course).

Few of the tasks are repeat of tasks you attempted in Lab 2/3, with a small change. So you need not explicitly work on Lab 2/3 tasks if you have not completed them.

Task 1: Realizing a 'switch' statement in RISC-V: Part II
In the previous lab, one of the tasks was -
Assume that there is a switch statement with three possible cases other than default. Write a RISC-V code to realize this switch.
a) We have studied the *jal* and *jalr* instructions in class. Using these instructions, redo the same task.
b) Now, after you complete (a), move the switch functionality to a procedure. Remember that x0, x1, x2 have their own specific purpose.

Task 2: In the previous lab, one of the tasks was -
Realizing a loop in RISC-V.
Using the control instructions we have studied in class, write a code that will perform the following tasks on elements in an array. For simplicity, consider an array of 10 elements in range [-50, 50]. As usual, preload the values of the 10 elements into the memory (Hint: Did you try `var1: .word 10, -20, 30, 40, 5` ?) and load one of the registers with the base address of this array.
1. Perform addition of all the elements.
2. Add those elements which are greater than a value, say, 5.
3. Consider only the positive elements which are below a value, say, 40.
4. Add only the negative elements.
5. Add all the elements which are greater than or equal to 35, considering them as unsigned.

In this lab, write a single code to perform one (selected as a number read from a memory location) of the five tasks, using a switch statement. Further, write each of the tasks as a procedure. You can reuse your code of previous lab.

Task 3: Procedure calling a procedure
Write a program that calls a procedure to return with the final grade for a subject (say, your favourite CA!). The procedure would in turn call other procedures like - proc_best4,

proc_best5, weighted_avg, grade, etc. Assume 7 theory assignements (pick best 4), 7 quizzes (pick best 5), 4 lab assignments (pick 3), 2 lab tests, 1 lab project, 1 mid term and 1 end term exam. Weightage of each - 5%, 10%, 5%, 10%, 10%, 25%, 35%, respectively. To make things interesting, let us assume we have only 10 (x0 - x9) registers at our disposal (in stead of 32!). Remember that x0, x1, x2 have their own specific purpose too. Consider just integers and neglect any decimals.

Task 4: Write a procedure "MEMCMP" for performing a byte-by-byte comparison of two sequences of bytes in the memory. The procedure should accept three input parameters in registers representing the first address (x5), the second address (x6), and the length of the sequences to be compared (x10). It should use a register (x11) to return the count of number of comparisons that are a mismatch.

Task 5: Write a procedure "ALLCAPS" that accepts a single parameter in a register (x13) representing the starting address STRNG in the memory for a string of ASCII characters in successive bytes representing an arbitrary collection of sentences, with a NULL control character at the end of the string. The task of the procedure is to scan the string beginning at address in x13 and replace every occurrence of a lower-case letter ('a' - 'z') with the corresponding upper-case letter ('A' - 'Z').

Task 6: Let's look at recursion now!
1. Write a recursive procedure for calculating factorial of a given number.
2. For better understanding of how recursion is different from iterative process, write iterative version of calculating factorial of a given number.

Task 7: "PALINDROME"
Write a procedure that determines whether an input string of characters is a palindrome or not. Input to the procedure is the starting address of the string (in register x15), and output is "yes" or "no" at the location 0x10000100.

Task 8: Fibonnaci series
Write a recursive procedure that generates 'n' Fibonnaci numbers. Give option to the user for seed values, in addition to 'n'.

Task 9: Bubble Sort
Write a recursive procedure to implement Bubble Sort algorithm. Input to this procedure is the starting address of the elements to be sorted (x10) and number of elements in another register (x11). Place the sorted elements in address given in register (x12). Note: Make sure that the control returns to FALL-THRU code with SP restored.

Task 10: Binary Search
Write a program to perform binary search for a given number 's', *recursively*. The program should store the location of 's' in x10 if present, -1 otherwise. Note: Make sure that the control returns to FALL-THRU code with SP restored.

Task 11: Once you are finished with *all* the above tasks, upload a single text file documenting your learnings and interesting aspects that you have noticed today. Write point-wise statements and not paragraphs. I will expect more insights/interesting aspects/learnings from your experiences from working with procedures and stack. Upload the file by 3rd Mar 2025 11.59AM.