CS 3432 – Computer Organization Author: Robert Alvarez

Lab 4.2 – RISC-V 32I Processor in Logisim

**Instructions:**

The assignment would be done with the same team as Lab 4.1. It is permitted to share ideas with other classmates BUT you must write your code.

**Introduction:**

[C expansion] During compilation time, programs get expanded into multiple expressions to try to achieve one-to-one correspondence with assembly instructions. For example, “1 << a1” doesn’t have an assembly instruction, so it gets expanded into “t1 = 1” and “t1 << a1” so we can do “ADDI t1, x0, 1” followed by “SLL a1, t1, a1”.

Another “program expansion” that happens concerns loops, branches, and function calls where labels and references to those labels are added. For example the following while loop while (\*a != '\0') ++a” can be translated into:

if (\*a == ‘\0’) goto end1;

start1:

a += 1;

if (\*a != ‘\0’) goto start1;

end1:

[C to Assembly] Once the program has been expanded, a sequence of assembly instructions that simulates the code is generated. For example, assuming that “char \*a” is inside register A0, the previous while loop can be translated into:

LBU a3, 0(a0)

BEQ a3, x0, end1

start1:

ADDI a0, a0, 1

LBU a3, 0(a0)

BNE a3, x0, start1

end1:

[Assembly to Machine Code] Then, the labels are translated into their respective numeric values based on their location. For example, the “end1” in “BEQ a3, x0, end1” gets a value of 16, and “start1” in “BNE a3, x0, start1” gets a value of -8. After all, the machine code is finally generated:)

The purpose of this lab is for you to see the entire process that happens between writing a program, specifically a C program, and running such a program. You would be able to test such a C program in just a terminal, Assembly in the RISC-V, and RISC-V Machine Code in your circuit from Lab 4.1.

Reminder:

From Lab 4.1 you were required to implement AND, OR, XOR, ANDI, ORI, XORI, ADD, SUB, ADDI, LW, SW, BEQ, BNE, BLT, and BGE instructions and given the option to implement SLL, SRL, SRA, SLLI, SRLI, SRAI, JAL, JALR, LH, LB, SH and SB as extra credit.

Task 1. [5 points] C to Assembly.

For this task, you are required to write multiple C programs along with their assembly translation. Between all programs combined, you should cover at least 70% of the instructions that you implemented in Lab 4.1. An example of what we are looking for in this part is in the “C\_to\_Assembly.docx” file.

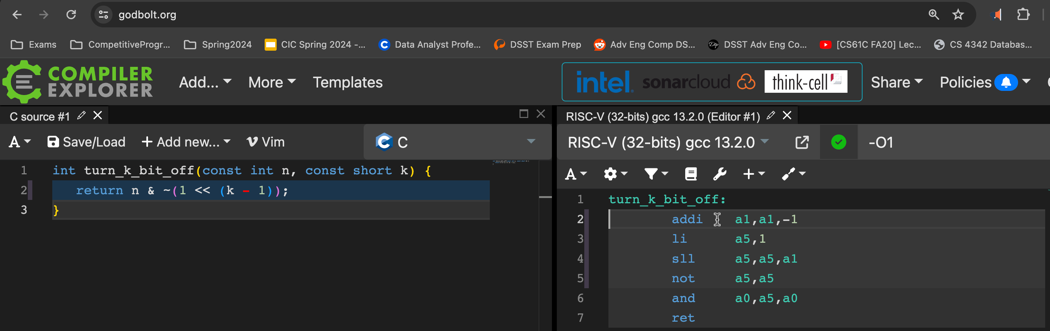
NOTES:

* DO NOT use the C to Assembly programs provided in “C\_to\_Assembly.docx” file.
* [5 extra pts] The “Memory Allocation” section is optional.
* You can use “[godbolt.org](http://godbolt.org/)” to translate your programs from C to Assembly.
* If you decide to translate using this website, you can compile using the optimization level 1 “-O1” to delete unnecessary access to the stack.

A screenshot of a computer

Description automatically generated

* Your C code in the first column and the Assembly instructions in the third column must be properly aligned.
* By putting your cursor on top of the assembly instructions in “godbolt.org” you can easily align them.



* [5 extra pts] No need to add the second column which is the C expanded version of the first column. But, if you do so, you will get 5 points of extra credit.

Task 2. [5 points] Running Assembly.

For each assembly code in task 1, you must put it into the “program.s” template file, run it, and see the output. For example, for the “Turn k bit off” it would look as follows:

A screenshot of a computer screen

Description automatically generated

The output would look like this:

A screenshot of a computer

Description automatically generated

Task 3. [5 points] Assembly to Machine Code.

This task consists of generating the Machine Code for each of your Assembly programs in task 1. The following repository can translate assembly into machine code: <https://github.com/RobertJAlvarez/RISCV_To_MachineLanguage>. Read the README.md inside the repository first to understand the overall flow. Then, you want to copy each of your assembly codes from task 1 into the test.asm, run “make run”, open the MCode.mc, copy the machine code and paste it into your own Assembly\_To\_MachineCode.docx file.

[5 extra pts] Then, for each new assembly instruction operation, deconstruct one machine code line that encode the instructions and check that every bit was set correctly.

Task 4. [10 points] Test the Machine Code in your circuit.

TODO

**Deadline (Blackboard):** May 3rd, 2024, by 11:59 pm.

1. C\_to\_Assembly.docx for task 1.
2. A PDF file with all outputs for task 2.
3. Assembly\_to\_MachineCode.docs for task 3.

* TODO: A template like task 1 will be given to you.

1. A PDF with all outputs from task 4.

**Grading:** (Total: 40 points)

Each task has a weight grade value.