Question and Answer #2

Topics

Lab 3 review and discussion.

Content

1. Understanding the C-Code
   1. Here is a graphical example of the calls and returns of a fibonacci(4) call.

A close up of a blackboard

Description automatically generated

1. Looking at the fibonacci Godbolt compiler output
   1. Important notes before you look at the compiler output:
      1. This is from a compiler that uses the frame pointer.
         1. Memory references will often be related to the frame pointer (s0).
         2. Frame point to the start of a stack frame and a compiler may use them to structure the stack like so:



* + - 1. Frame pointers are useful b/c the stack pointer may change during the course of a procedure call, but the frame pointer will remain constant making simpler addressing.
      2. If stack pointer only changes at the very beginning and end of a procedure the same utility can be had without the need for the frame pointer (as in the text’s examples).
    1. The compiler is heavily dependent on pseudocodes. Here are the commonly translations of the pseudocodes used in the compiled code:
       1. **mv rd, rs1** - move rs1 contents to rd
          1. can be satisfied with addi rd rs1 0
       2. **call label** - jump to label and link to ra (x1)
          1. can be satisfied with jal ra, label
       3. **li rd imm** - load an immediate
          1. can be satisfied with addi rd zero imm
       4. **bnez rs1, label** - branch to label if rs1 != 0
          1. can be satisfied with bne zero, rs1, label
       5. **jr label** - jump to a label and toss out ra
          1. can be satisfied with jal x0, label
    2. Both the original and annotated RISC-V code compiled with the Godbolt Clang compiler are shown below.
       1. It starts with the main procedure which calls fibonacci(4).
       2. To run the code in the Venus simulator, paste the code into the editor, Assemble & Simulate, and place a break point immediately after the fibonnaci call in main.
       3. Stepping through the code can show valuable insight into stack management and register contents.

