Craig Shaffer

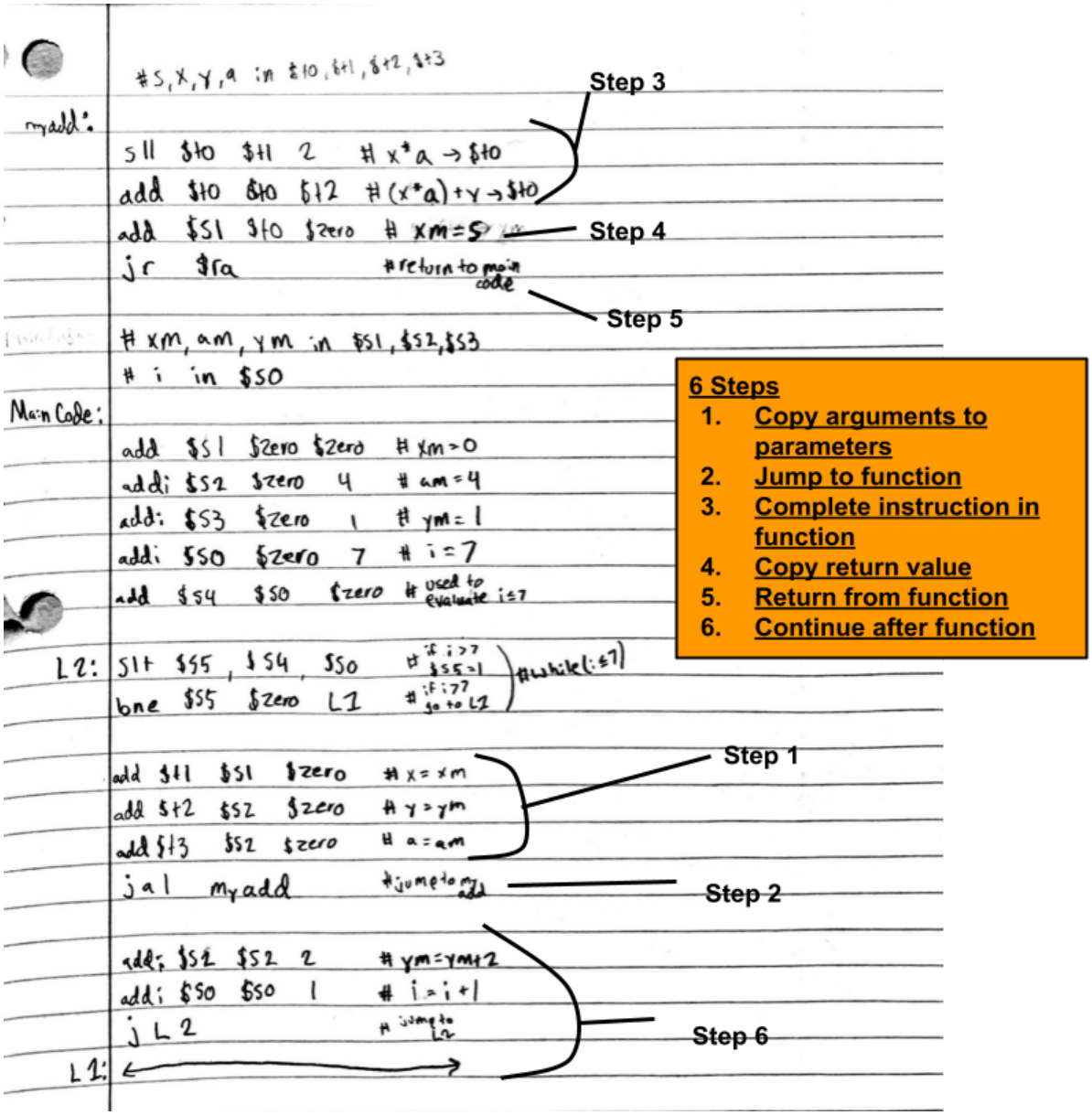
Dr. Swami Ponpondi

CPSC 300

4-6-2022

Homework 10

1. Write the MIPS code for the below code. Identify the 6 steps discussed in the class by adding comments. Assume the variables are in appropriate registers (caller, callee protocols). [40]



2. The below is a recursive function. Are there any difficulties in writing the MIPS code? Explain your answer by writing the MIPS code, describing why the code may not work. How is the code for recursive functions handled in the textbook? NOTE: You must read the textbook to answer this question. Look into stack pointer register and its role in MIPS code. [10]

A difficulty in writing the code is that there is nowhere that we are specifically supposed to assign our function call to. Like we call myfunc(3), but its result has no specified location to store it into (step 4: copy return value). In the code below, I'm storing the value in $v0 (which is a value register) like the book does on page 101. Another difficulty is the use of stacks in the book. Since we haven't had much practice with the idea it is somewhat difficult, but the book's explanation of $sp (stack pointer) makes it easier to comprehend. By storing values in the stack pointer, we are saving them so we can use their registers for something else. When we need those values again, we can simply load them back from the stack pointer. In the book, recursive functions specifically make use of this because they have to store return addresses and values for when they get back to the original call so they can return to the correct location in the main code.

The code may not work because we don't know where to assign the result of our function call since its just "myfunct(3)" instead of something like "variable1=myfunct(3)". Since we aren't storing it somewhere for usage in the main code, it might not work as we intend it to because its just being called and all the work it does is basically for nothing. Below is my MIPS code for the function.

```
myfunc:                 #i in $a0
        addi $sp, $sp, -8       #adjust stack for 2 items
        sw $ra, 4($sp)          #save the return address
        sw $a0, 0($sp)          #save argument i

        slt $t0, $zero, $a0     #check if 0 < i. If it is $t0=1
        bne $t0, $zero, L1      #if $t0 is 0 jump to L1

        addi $v0, $zero, 1      #return 1
        addi $sp, sp, 8         #pop 2 items from stack. $ra and $a0 don't change so we dont have to
                                #load them back in
        jr $ra                  #return to caller
  L1:   addi $a0, $a0, -1       #i-1 in $a0
        jal myfunct             #call function with i-1

        lw $a0, 0($sp)          #return from call, restore i
        lw $ra, 4($sp)          #restore $ra
        addi $sp, sp, 8         #pop 2 items from stack

        addi $v0, $a0, 1        #return 1 + myfunct(i-1)
        jr $ra                  #return to caller
main code:
        addi $s0, $zero, 3      #store three in $s0
        add $a0, $s0, $zero     #copy arguments to parameters

        jal myfunct             #myfunct(3)
```