1 Summary

1. Symbol tables
2. Outputting bytes
3. Assembly errors
4. Outputting instructions

2 Problems

1. Construct the symbol table for the following MIPS assembly program.

   ```
   begin:
   label: beq $0, $0, after
   jr $4

   after:
   sw $31, 16($0)
   lis $4
   abc0: abc1: .word after

   loadStore:
   lw $20, 4($0)
   sw $20, 28($0)

   end:
   ```

2. Write pseudocode for a function called `output_word` that takes a 32-bit integer as input and outputs each of its four bytes to standard output.

   You can assume a function called `output_byte` is available. This function takes an integer as input and first checks if the integer is small enough that it can be represented in 8 bits. If so, it outputs the corresponding byte; otherwise it produces an error.

   How would you use this above function (in conjunction with the symbol table) to assemble the `.word after` directive in the above program?

3. Identify the errors in the following assembly language program.
4. Write a function called `assemble_add` that assembles instructions of the form `add $d, $s, $t`. It should take three integer parameters (the numbers $d$, $s$ and $t$) and return a single integer representing the binary encoding of the add instruction.

Can we generalize the `assemble_add` function for other instructions? How?