Question 1 (30 points): You are provided with the listing of an assembly program in MIPS that contains the code for two versions of a routine that computes the number of characters in a null-terminated string: `StringLenByte` and `StringLenWord`.

Assume that arithmetic and logic instructions (`add`, `addi`, `li`, `move`, `and`, `sll`) each take one cycle to execute; branches and jumps each take three cycles to execute, and loads (`lw`, `lb`) each take ten cycles to execute.

You are asked to answer the following questions about these routines.

a. (**5 points**) How many instructions are executed by each of the subroutines if `$a0` contains the address of `StringA` when the routine is called?

b. (**5 points**) What are the values, expressed in hexadecimal, in registers `$t0`, `$t1`, and $t2 at the end of the execution of `StringLenWord` if this routine is called with the address of StringA in `$a0`? (in the ASCII code: 'a' = 0x61, 'c' = 0x63, 'l' = 0x6C, 's' = 0x73)

c. (**10 points**) Assume that `$a0` contains the address of a null-terminated string that is formed by 255 characters when each subroutine is called. What is the average number of clock per instruction (CPI) for each subroutine? Show your calculations.

d. (**5 points**) For the 255-character null-terminated string, which subroutine is faster? By how much? Show your calculations.

e. (**5 points**) The subroutine `StringLenWord` is written for a little-endian machine. Would this routine work in a big-endian machine? If yes, explain why the endianness of the machine does not affect how the routine works. If no, explain which lines of the routine need to be changed and how they should be changed to create a big-endian version of the routine.