6. In this question you are required to make calculations regarding the processor's speed. For this question, use the MIPS assembly program from the previous section.

   (a) (2 points) How many instructions are executed until the program finishes?

   > **Solution:** There are two initial instructions, and the loop contains 7 instructions which is executed 8 times, in addition there is one last `beq` command that will be executed after the last loop. $N_{instructions} = 2 + (7 \times 8) + 1 = 59$ (56, or 58 could also be accepted)

   (b) (1 point) Assume that you are using a single cycle microarchitecture running at 1 GHz. If the CPI (Cycles Per Instruction) is 1, how long will it take for the program to finish execution?

   > **Solution:** 59 instructions x 1 CPI = 59 clock cycles. At 1 GHz, 1 clock cycles is 1 ns, so the entire program will execute in 59 ns.

   (c) (2 points) In a second microarchitecture, due to a very slow memory, every *load word* (`lw`) instruction requires 10 clock cycles. Calculate the number of cycles needed to execute the entire program.

   > **Solution:** The loop contains 2 `lw` instructions each take 10 clock cycles. The remaining 5 instructions take 1 clock cycle. Every loop takes 25 clock cycles. There are 8 iterations of the loop plus two additional initial instructions and one last `beq`: $N_{cycles} = 2 + (8 \times (5 \times 1 + 2 \times 10) + 1 = 203$ (200, or 202 could also be accepted)

   (d) (1 point) For this second microarchitecture, what is the average CPI? (*approximate numbers ±10% are ok*)

   > **Solution:** There are 59 instructions that are executed in 203 clock cycles. $CPI = 203/59 = 3.44$. Acceptable are also approximate calculations $CPI = 200/60 = 3.33$ or $CPI = 210/60 = 3.5$

(e) (4 points) For the second microarchitecture with the slow memory, what can be done in both the code and the processor to improve the speed of this particular program other than using a faster memory. Write two short suggestions that would have the largest impact and briefly explain why that would improve the performance.
Note: The result of the program should depend on the two values stored in the memory, these can not assumed to be constants.

Solution:

- The most obvious problem is the slow memory access. Introducing a proper cache could alleviate this problem. The first iterations would still be slow, but the remaining accesses would be fast.
- The program could be written more efficiently. The loop essentially multiplies the sum of the two values in memory by eight. Instead of a loop, a shift left operation could be used.

```
lw   $t1, 0x4($0)
lw   $t2, 0x24($0)
add  $s0, $t1, $s2
sll  $s0, $s0, 3
```

- Standard pipelining will not work in this architecture, there are two consecutive memory accesses which are the problem, all other instructions can already be calculated in a single cycle.
- Using a better technology would increase the speed. However, this is not one of the easiest solutions. The first two solutions are clearly better suited.