Question 1 (25 points): In the reverse engineering of a computer application, an important task is to write the source code for a corresponding segment of assembly code. You are working for a security agency and you have been asked to provide source-level code for the function `enigma` whose assembly code is below. The code is printed with line numbers to facilitate referencing to instructions in your answer.

```
(1) enigma:
(2)       add  $t0, $zero, $zero
(3)       slt  $t1, $t0, $a2
(4)       beq  $$t1, $zero, label_one
(4') label_two:
(5)       sll  $t2, $t0, 2
(6)       add  $t3, $a1, $t2
(7)       lw   $t4, 0($t3)
(8)       sll  $t5, $t4, 2
(9)       add  $t6, $a0, $t5
(10)      lw   $t7, 0($t6)
(11)      lw   $t8, 4($t3)
(12)      sll  $t9, $t8, 2
(13)      add  $t1, $a0, $t9
(14)      sw   $t7, 0($t1)
(15)      addi $t0, $t0, 1
(15')     blt  $t0, $a2, label_two
(16) label_one:
(17)      jr $ra
```

a. (**10 points**) Assume that this function follows the MIPS procedure-calling conventions. How many parameters does the function `enigma` has? Give a name for each parameter. You you will use these names in your source code. Also, indicate the type of the parameter (is it an address or a value? In the case of a value, can you say anything about the number of bits?) Justify your answer.

b. (**10 points**) How many memory loads and how many memory stores are executed by `enigma`? Your answer can be in terms of one or more of the parameters of the function.

c. (**5 points**) Write C-style source code that leads to the generation of the assembly code above for `enigma`.