Lines 1 to 13 are used for defining the global variables and the format strings are created to receive the inputs and print the outputs.

**pushq %rbp:**

Used to push the frame pointer to the stack, which saves it. As %rbp is a callee saved register, its value needs to be preserved throughout the function call. Hence it is pushed to the stack. This saves the caller's value for %rbp into the callee's stack.

**movq %rsp, %rbp:**

Used to move the contents from the source register (%rsp) to the destination register(%rbp). This would be represented as %rbp = %rsp. This saves the current stack pointer in %rbp. The q in movq represents a quad word (the operation is taking place using long datatype). This operation is direct register reference.

**movq $format1, %rdi:**

This operations puts the constant/contents of the format1 string (“a=”) into the first argument register. %rdi gets whatever the format1 string points to. %rdi is pointing to the contents of the format1 string.

**movq $0, %rax:**

This operation puts the constant 0 into the %rax register (return value register). This can be seen as initializing the return value to 0. As printf takes in a variable number of arguments you have to put a 0 in %rax.

**call printf:**

This operation goes to the printf function to print the given format string to the terminal.

**movq $format2, %rdi:**

This operations puts the constant/contents of the format2 string (“%ld”) into the first argument register. %rdi gets whatever the format2 string points to. %rdi is pointing to the contents of the format2 string.

**movq $a, %rsi:**

This operation moves the address of a into the rsi register which receives the value of a using scanf.

**movq $0, %rax:**

This operation puts the constant 0 into the %rax register (return value register). This can be seen as initializing the return value to 0. As scanf takes in a variable number of arguments you have to put a 0 in %rax.

**call scanf:**

This operation goes to the scanf function to receive the input value and store it in the corresponding registers and variables.

**movq $format3, %rdi:**

This operations puts the constant/contents of the format3 string (“a^2 is %ld\n”) into the first argument register. %rdi gets whatever the format3 string points to. %rdi is pointing to the contents of the format3 string.

**movq $a, %rsi:**

This operation moves the address of a into the rsi register so receive its actual value during multiplication.

**movq (%rsi), %rsi:**

The memory location pointed to by rsi (contents of address of a) is moved to rsi. The value of rsi is replaced by the value it points to in memory (the value at address of a). The address of a contains the value of a, which moves into rsi.

**imluq %rsi, %rsi:**

This operation multiplies the value stored in rsi with itself, giving the square. The result is stored in %rdx:%rax

**movq $0, %rax:**

This operation puts the constant 0 into the %rax register (return value register). This can be seen as initializing the return value to 0. As printf takes in a variable number of arguments you have to put a 0 in %rax.

**call printf:**

This operation goes to the printf function to print the given format string to the terminal.

**leave:**

This operation sets %rsp to %rbp and pops the top of the stack into %rbp (since it is callee-saved). This ensures that no matter what you do to the stack pointer in the function body, you will always return it to the right place when you return.

**ret:**

Pops the return address from the stack and jumps there.

Note – line 32, Memory location pointed to by rsi (address of a) is moved to rsi. Value of rsi is replaced by the value it points to in memory (the address of a). The address of a contains the value of a, which moves into rsi.