EXPERIMENT 7:

```
def expand_second_level_macro(instruction):
    # Second level macro expansion
    if instruction.startswith("TILAK"):
        arg = instruction.split()[1]
        return ["ADD " + arg, "MUL " + arg]
    else:
        return [instruction]
def expand_first_level_macro(input_source, first_level_macro_definition):
    # First level macro expansion
    expanded_source_code = []
    macro_calls_count = 0
    macro_instructions_count = 0
    for line in input_source.split('\n'):
        if line.strip().startswith("RAHUL"):
            macro_calls_count += 1
            macro_instructions = first_level_macro_definition.split('\n')[1:-1]
            macro_instructions_count += len(macro_instructions)
            for macro_instruction in macro_instructions:
                expanded_source_code.extend(expand_second_level_macro(macro_instruction))
        else:
            expanded_source_code.append(line)

    return expanded_source_code, macro_calls_count, macro_instructions_count
# Input source code with First level macro calls
input_source_code = """MOV R
AND R
RAHUL
MUL 88
HALT"""
# First level macro definition "RAHUL"
```

```
first_level_macro_definition = """MACRO
RAHUL
SUB R
TILAK 77
MUL R
TILAK 99
MEND"""
# Second level macro definition "TILAK"
second_level_macro_definition = """MACRO
TILAK &ARG
ADD &ARG
MUL &ARG
MEND"""
# Expand first level macro
expanded_source_code, macro_calls_count, macro_instructions_count =
expand_first_level_macro(input_source_code, first_level_macro_definition)
# Expand second level macro
final_expanded_source_code = []
for instruction in expanded_source_code:
    final_expanded_source_code.extend(expand_second_level_macro(instruction))
# Calculate statistics
total_instructions = len(input_source_code.split('\n')) - macro_calls_count +
macro_instructions_count
# Output final expanded source code
print("Final Expanded source code (after second level macro expansion):")
for instruction in final_expanded_source_code:
    print(instruction)
# Output statistics
print("\nStatistical output:")
print("Number of instructions in input source code (excluding Macro calls) =",
len(input_source_code.split('\n')) - macro_calls_count)
print("Number of Macro calls at first level =", macro_calls_count)
```

print("Number of instructions & other macro calls defined in the first level Macro call =", macro_instructions_count)

print("Total number of instructions in the final expanded source code =", total_instructions)

OUTPUT:

```
Final Expanded source code (after second level macro expansion):
MOV R
AND R
RAHUL
SUB R
ADD 77
MUL 77
MUL R
ADD 99
MUL 99
MUL 88
HALT

Statistical output:
Number of instructions in input source code (excluding Macro calls) = 4
Number of Macro calls at first level = 1
Number of instructions & other macro calls defined in the first level Macro call = 5
Total number of instructions in the final expanded source code = 9
```