Macroinstructions in the MiniMIPS assembly language

A *macroinstruction* or just *macro* is a mechanism to a give a name to a frequently-used sequence of instructions. This way, we do not need to specify the sequence in full each time we need to use it.

The general format defining a macro is as follows.

```
.macro  name(argument list)
# you can use basic instructions and pseudoinstructions in a marco.
…
…
.end_macro
```

While *pseudoinstructions* are incorporated in the design of the MiniMIPS, macros are user-defined. From its appearance, a macro is more like a *procedure* but it is replaced by the sequence of instructions contained it when assembled by the MiniMIPS assembler. After a macro has been replaced by its sequence of instructions, no trace of the macro itself remains in the program.

Look at the following macro, which determines the largest of the values in three registers, a1, a2 and a3, and put the result into the fourth register, m.

```
.macro max3(m, a1, a2, a3)      # macro and arguments
move m, a1                      # assume (a1) is the largest
bge m, a2, +4                   # if (a2) is not larger, ignore it
move m, a2                      # else set m = (a2)
bge m, a3, +4                   # if (a3) is not larger, ignore it
move m, a3                      # else set m = (a3)
.end_macro
```

You can see that the macro uses pseduoinstructions. In your program, then you can use it as follows.

```
…
# Determine the largest value of three registers $a0, $a1, $a2
# put the result into $v0
max3 $v0, $a0, $a1, $a2
...
```