Volatile vs. non-volatile resources

- **volatile** resources are represented by those registers that **the calling convention is defining them as belonging to the called subroutine**, thus, the caller being responsible **as part of the call code** to save their values (if the called subroutine is using them) and after that, at the end of the call to restore the initial (old) values. So: who is saving the volatile resources ? **The caller** (as part of the call code) . Who is restoring in the end those values ? Also **the caller** <u>but NOT as part of a certain call/entry or exit code</u>. Just restore them after the call in the regular code as a mandatory responsibility.

- **non-volatile** resources are any memory addresses or registers which do not belong explicitly to the called subroutine, but if this one needs to modify those resources, it is necessary that the called subroutine to save them at the entry as part of the entry code and restore them back at exit, as part of the exit code. So: who is saving the non-volatile resources ? **The callee** (apelatul = the **called** subroutine, as part of the entry code) . Who is restoring in the end these values ? Also **the callee** (as part of the exit code).