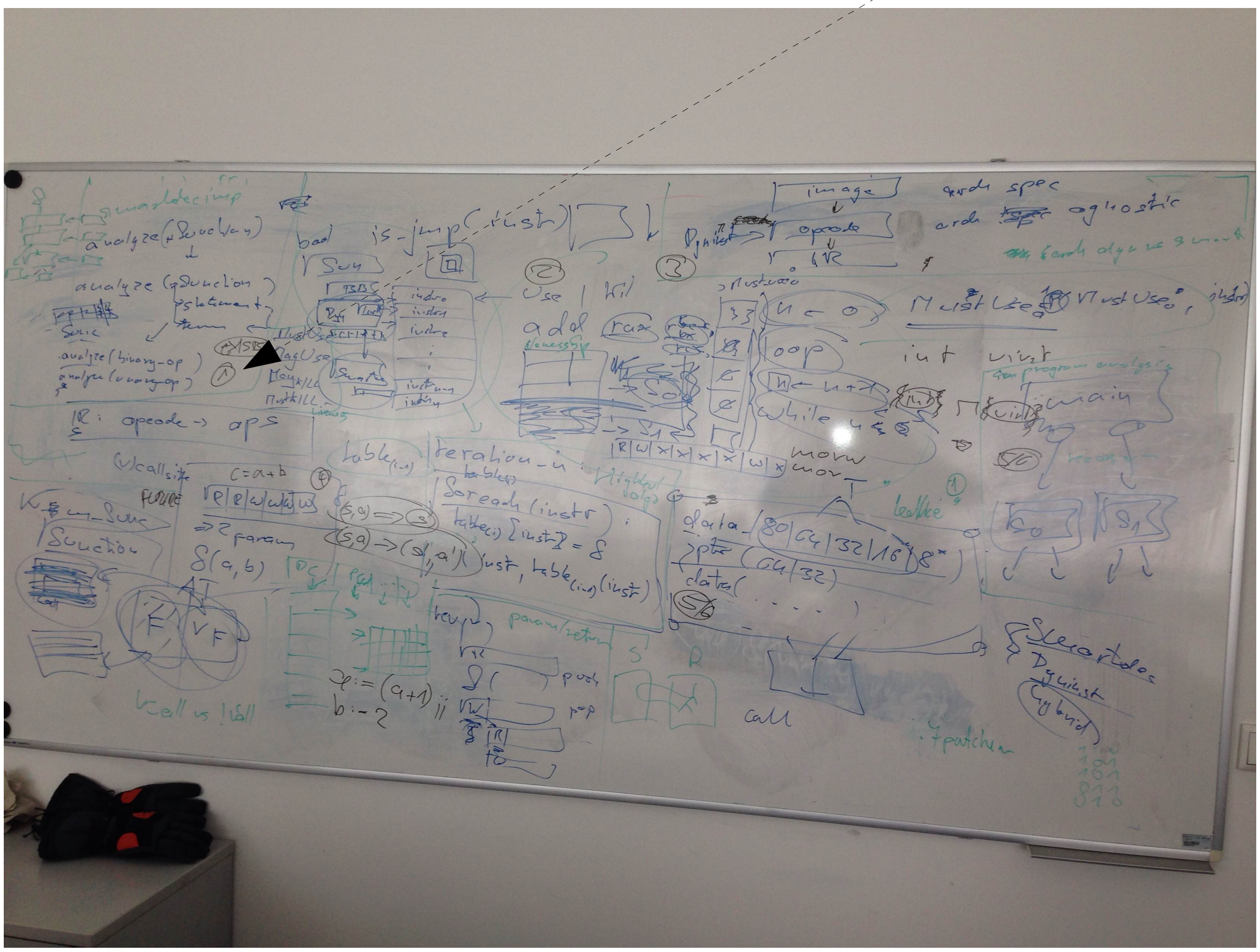
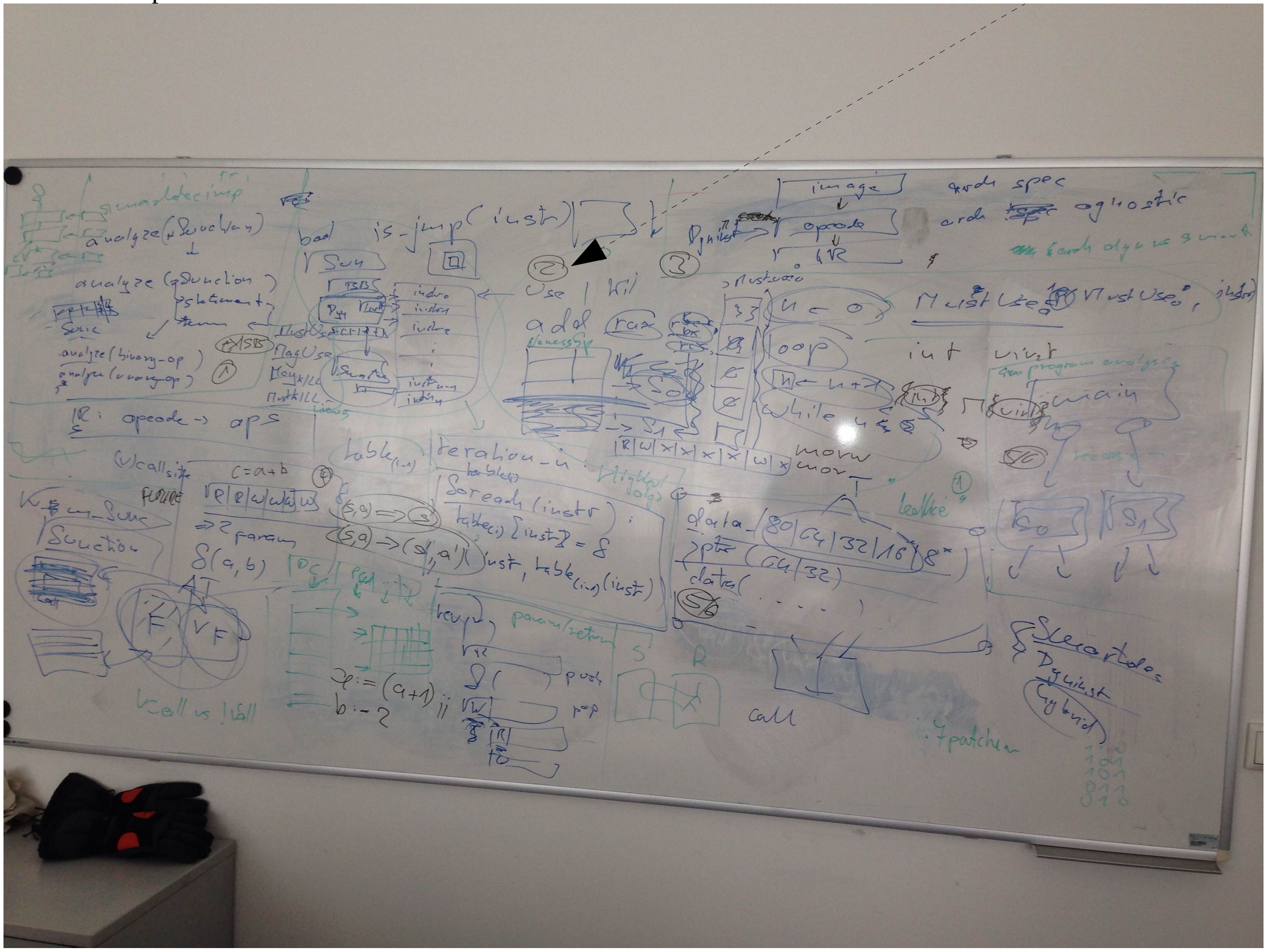
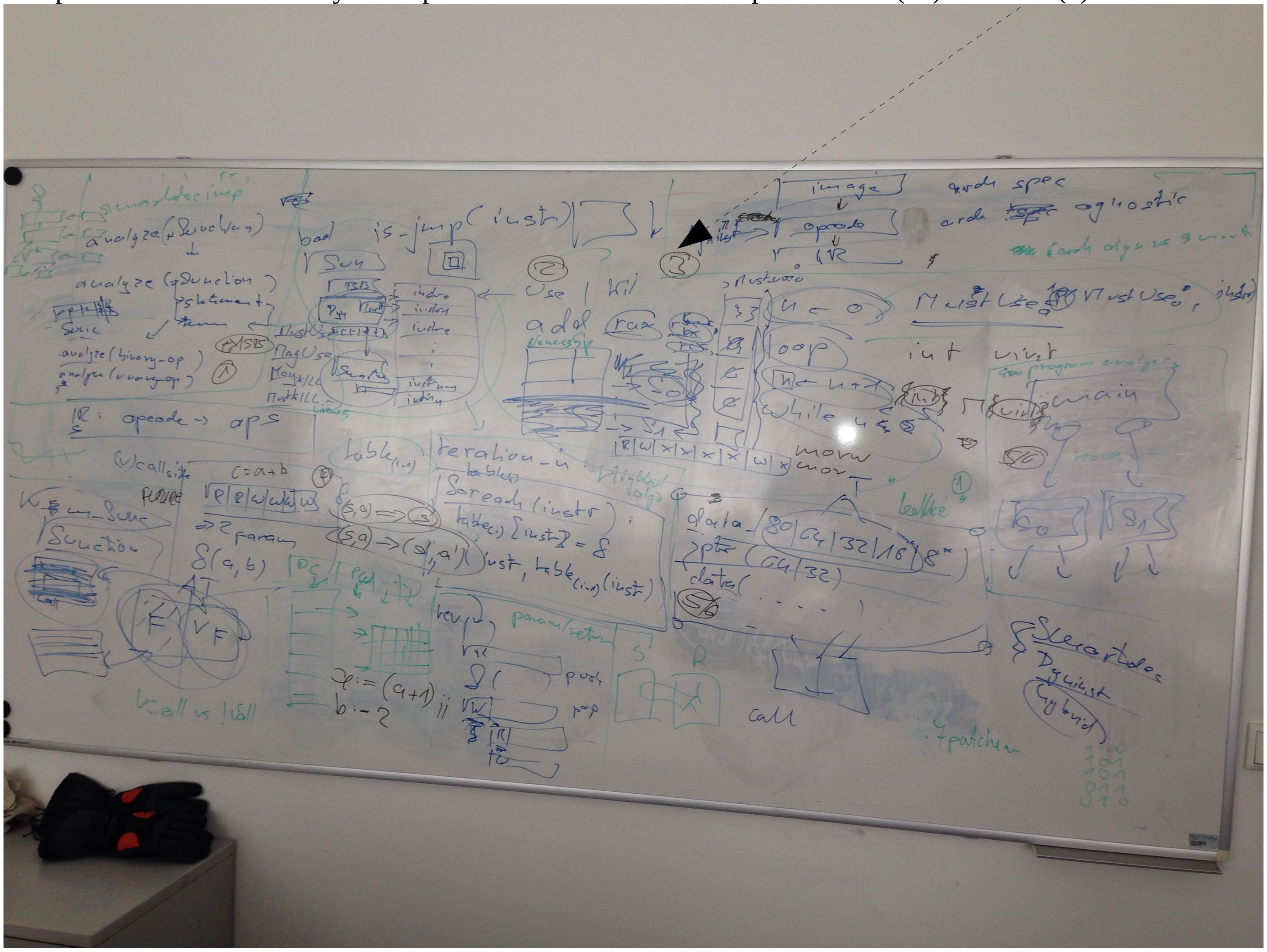
Step1: Analyze function, statement and term as it is done in SmartDec, see (1) in the left upper corner of fig 1.



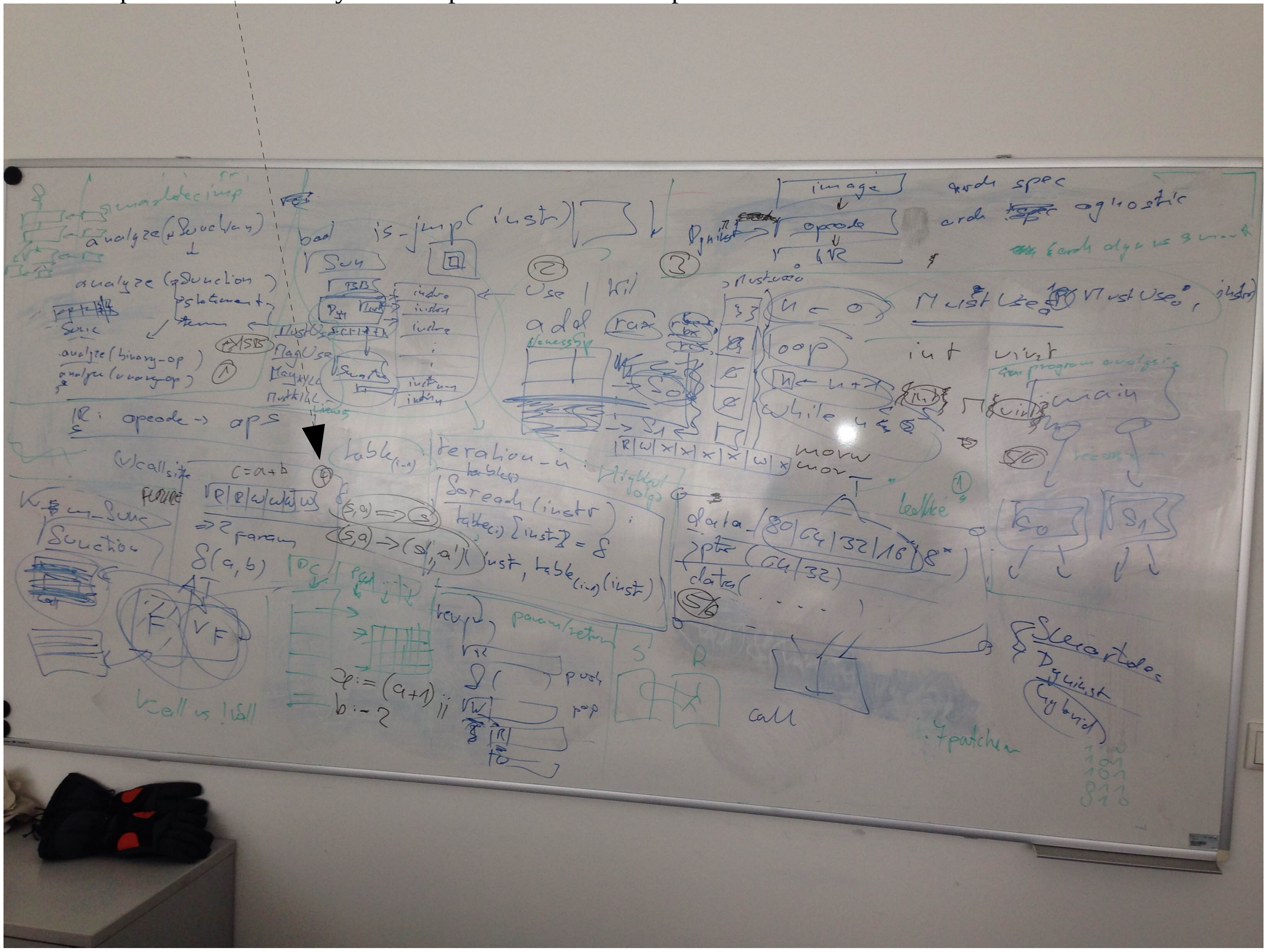
Step 2: The Fixed Point algorithm with predicates and liveness analysis implementation (2). The predicates are the most important issue here.



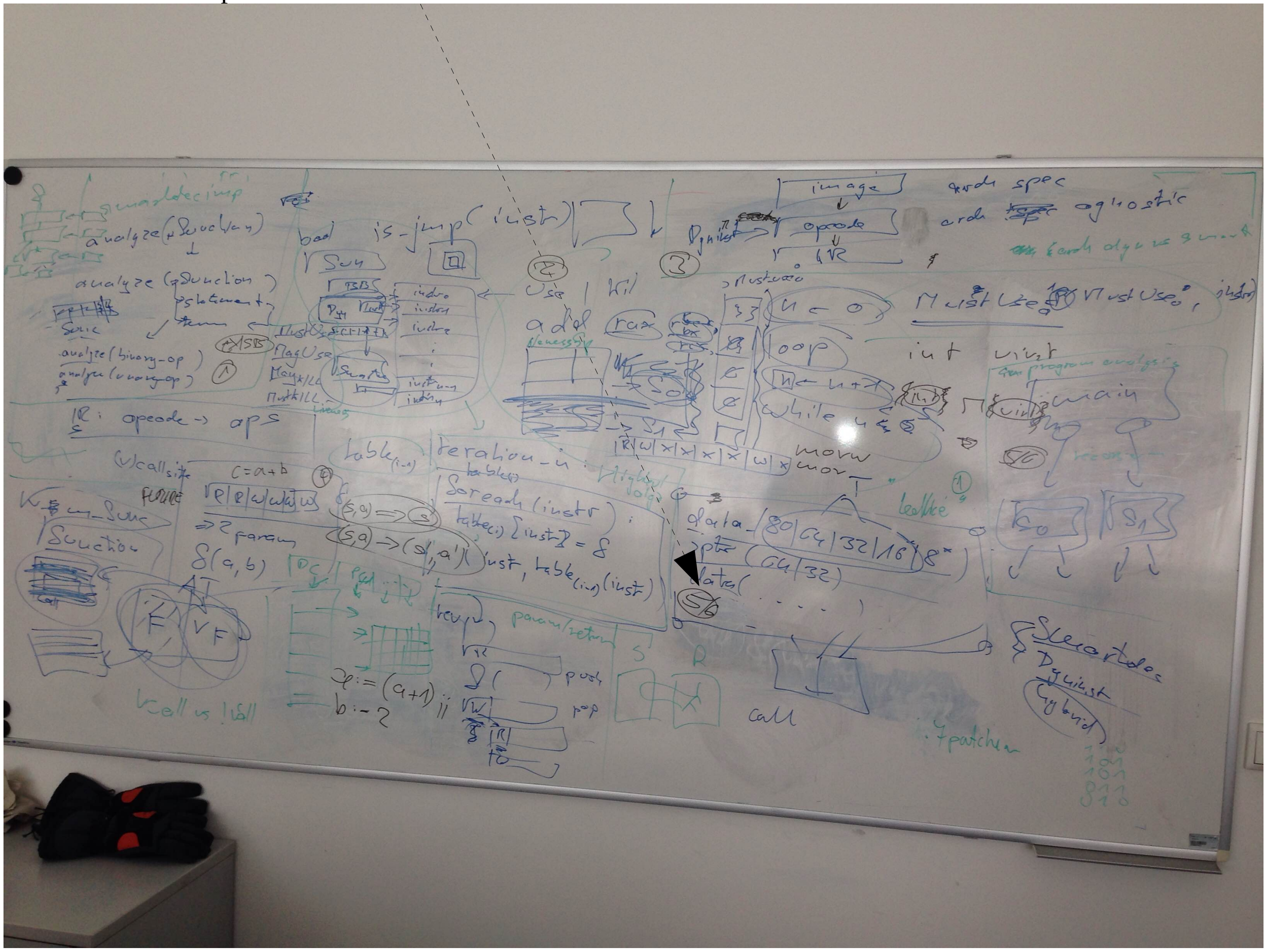
Step 3: Conversion of the Dyninst opcode to the Indermediate Representation (IR) available (3) in SmartDec.



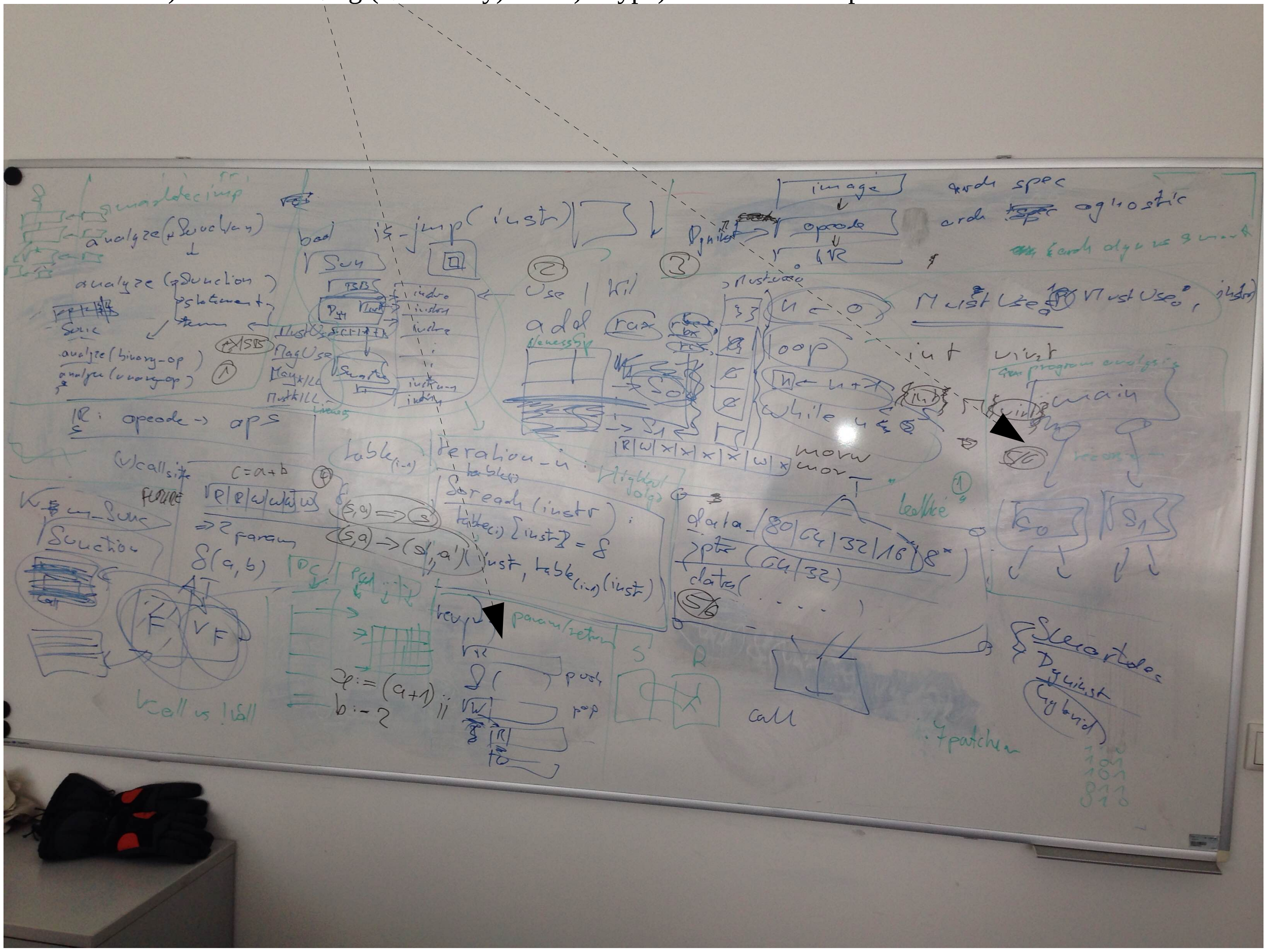
Step 4: Labeling (4) with MustUse, MayUse, MayKill, MustKill the mask containg the 6 registers for each function present in the binary. This step is the result of Step 2.



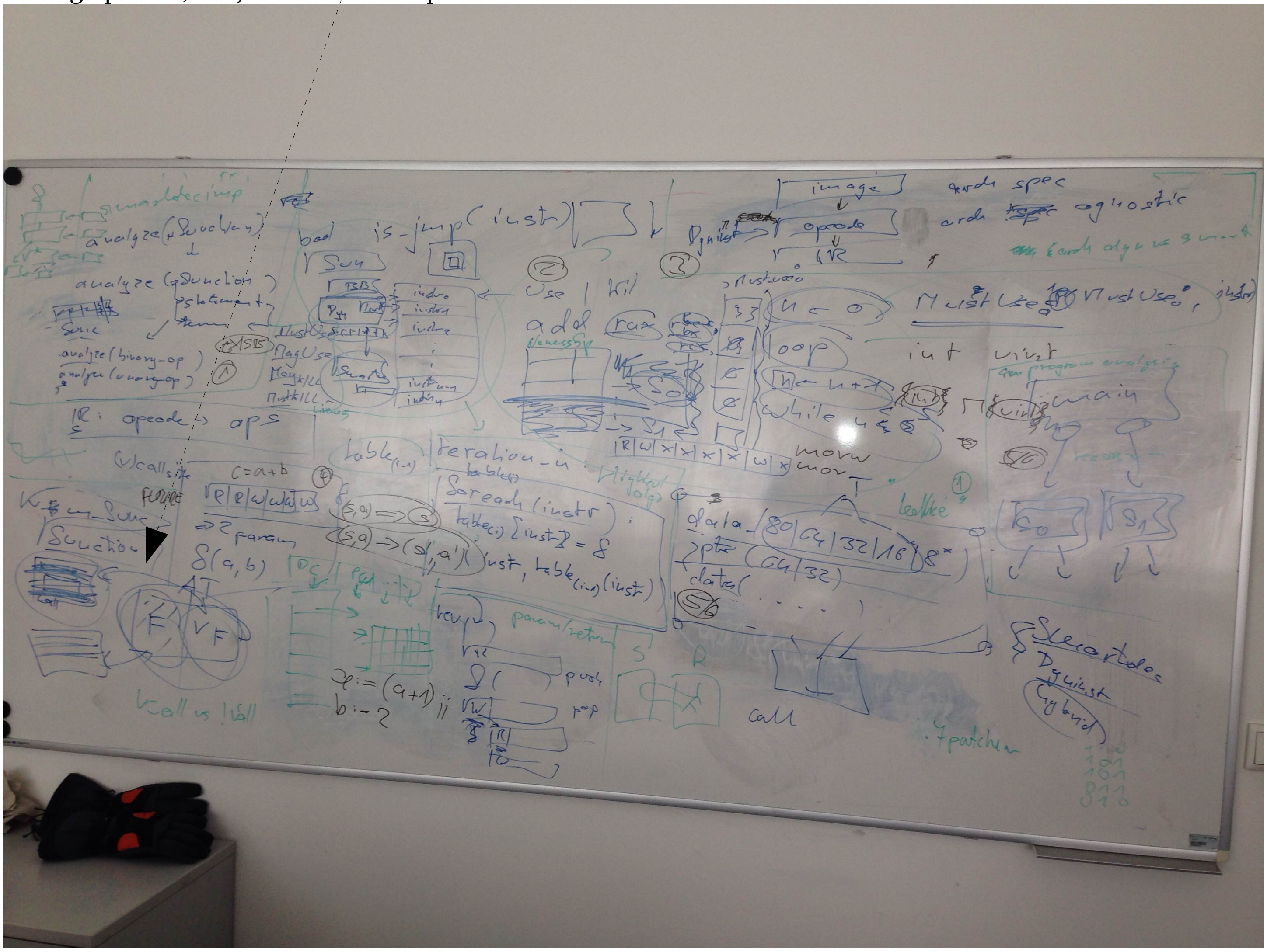
Step 5: Definition of the Lattice (5) (this should be straight forward) inside the variable type analysis. This will be used inside Step 2.



Step 6: Whole program (6) analysis with asignment of function parameter registers (the 6 registers from the mask to to the local variables inside each function such that more the function parameters can be more constrained a) with numbering (how many) and b) a type). This would help to reduce candidate callee locations.



Step 7: Nice to have feature (7). A differentiation between the types of indirect calls (vcall through object, vcall though pointer, etc.). This would help to reduce candidate caller locations.



Step 8. Bianry Patching, This was not discussed last time. Step 8, has to be performed after the whole program analysis. Checks have to be inserted at the caller/callee pairs.