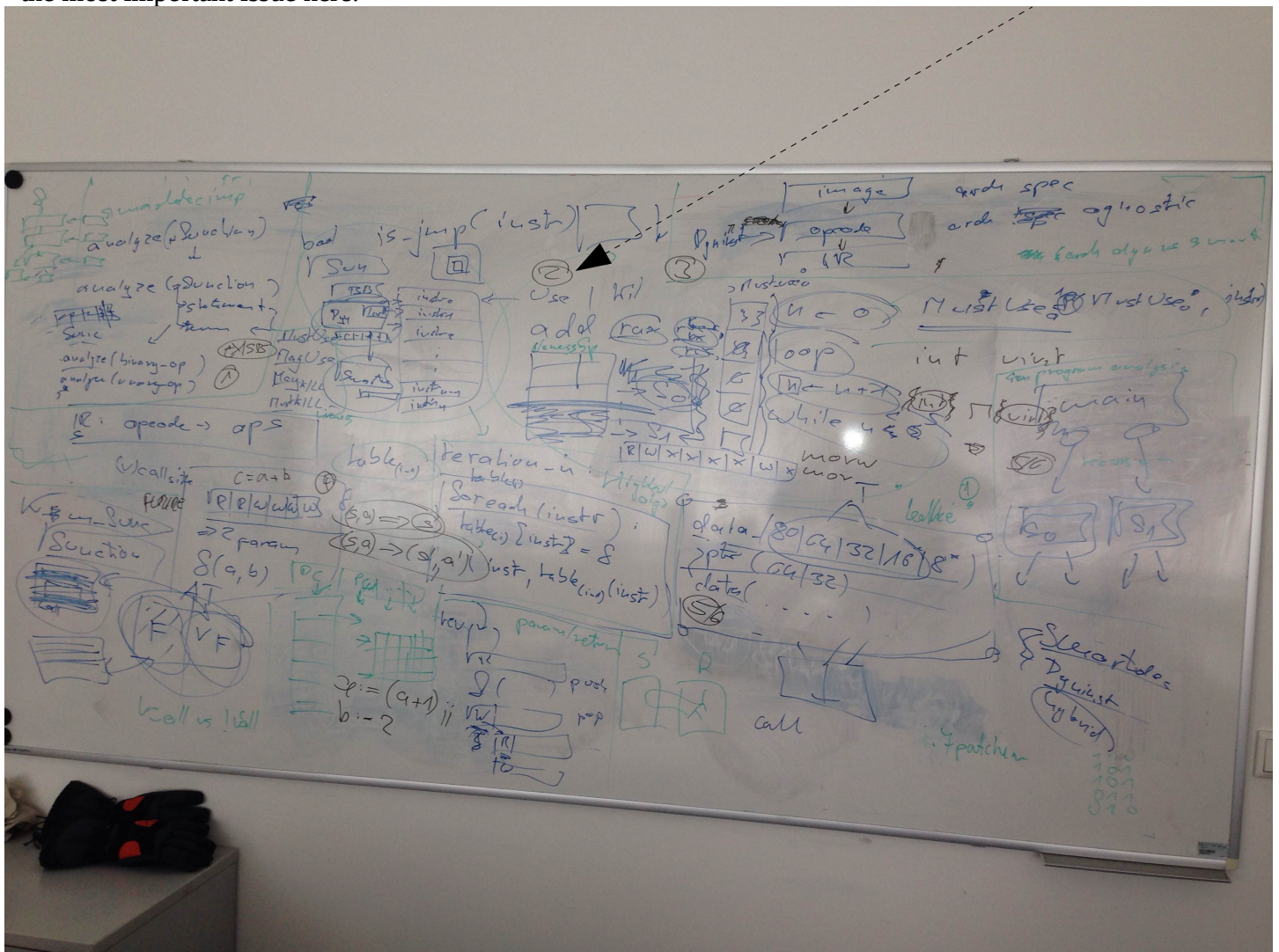
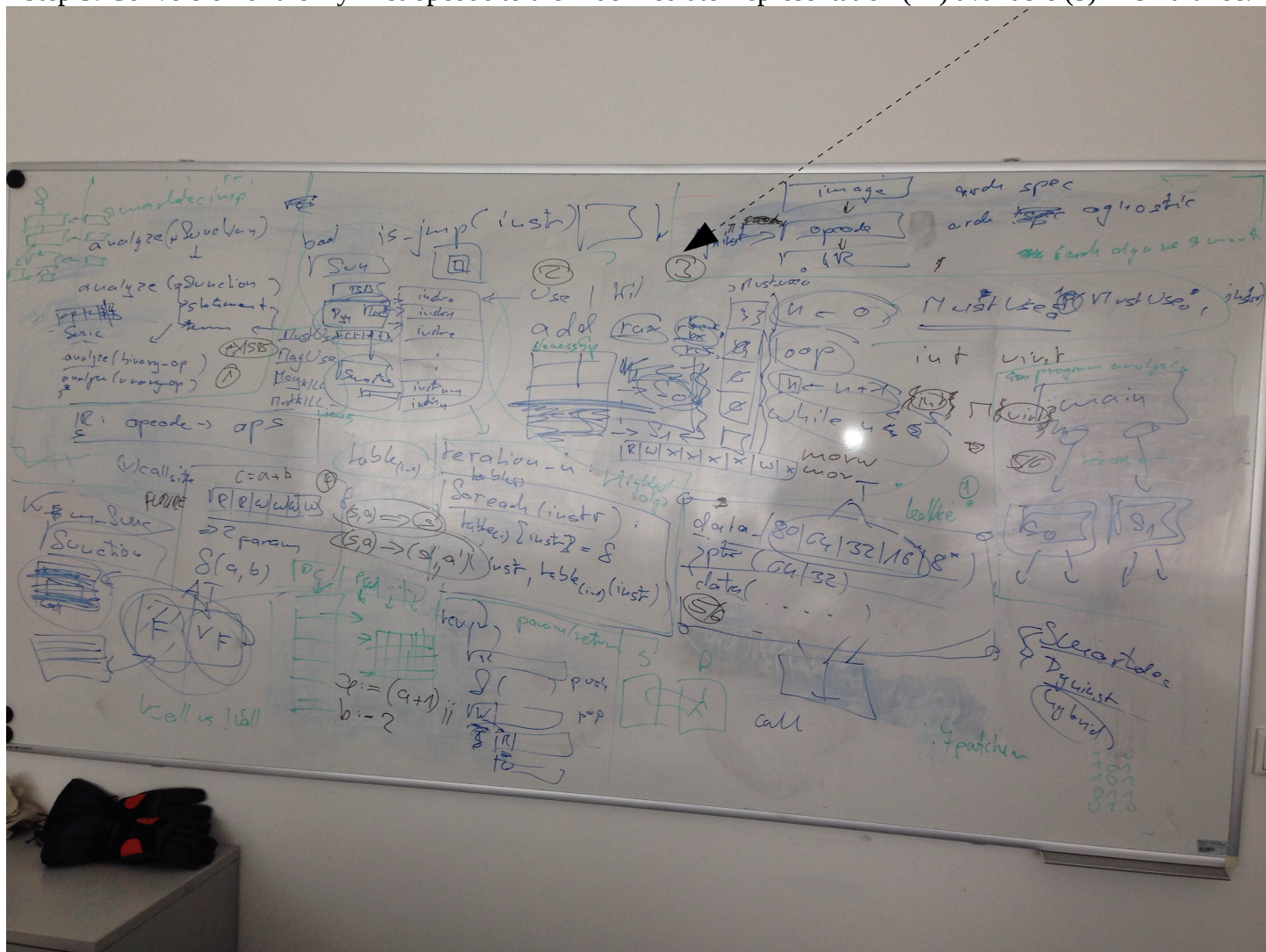


Step1: Analyze function, statement and term as it is done in SmartDec, see (1) in the left upper corner of fig 1.

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



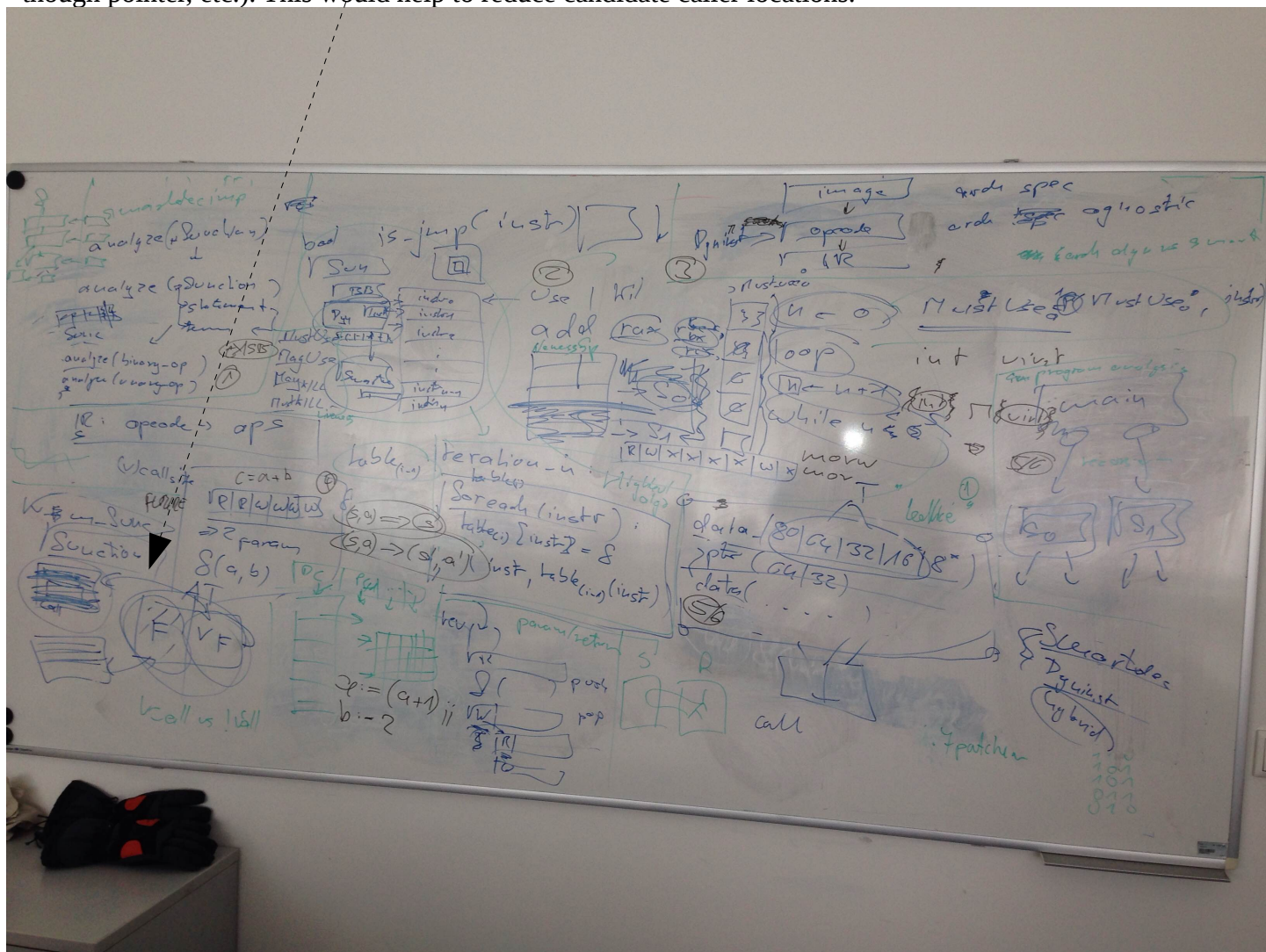
Step 3: Conversion of the Dyninst opcode to the Intermediate Representation (IR) available (3) in SmartDec.



The whiteboard contains the following handwritten content:

- Top Left:** A small diagram showing a sequence of instructions with arrows indicating flow. Below it, the text "analyze (function)" is written.
- Top Center:** A diagram of a basic block. It is a rectangle divided into two parts. The top part is labeled "header" and contains "entry" and "exit" nodes. The bottom part is labeled "body" and contains a list of instructions: "add r1, r2, #1", "sub r3, r4, #2", "mul r5, r6, #3", "div r7, r8, #4", "and r9, r10, #5", "or r11, r12, #6", "xor r13, r14, #7", "shl r15, r16, #8", "shr r17, r18, #9", "lshl r19, r20, #10", "lshr r21, r22, #11", "rshl r23, r24, #12", "rshr r25, r26, #13", "and r27, r28, #14", "or r29, r30, #15", "xor r31, r32, #16", "shl r33, r34, #17", "shr r35, r36, #18", "lshl r37, r38, #19", "lshr r39, r40, #20", "rshl r41, r42, #21", "rshr r43, r44, #22", "and r45, r46, #23", "or r47, r48, #24", "xor r49, r50, #25", "shl r51, r52, #26", "shr r53, r54, #27", "lshl r55, r56, #28", "lshr r57, r58, #29", "rshl r59, r60, #30", "rshr r61, r62, #31", "and r63, r64, #32", "or r65, r66, #33", "xor r67, r68, #34", "shl r69, r70, #35", "shr r71, r72, #36", "lshl r73, r74, #37", "lshr r75, r76, #38", "rshl r77, r78, #39", "rshr r79, r80, #40", "and r81, r82, #41", "or r83, r84, #42", "xor r85, r86, #43", "shl r87, r88, #44", "shr r89, r90, #45", "lshl r91, r92, #46", "lshr r93, r94, #47", "rshl r95, r96, #48", "rshr r97, r98, #49", "and r99, r100, #50".
- Top Right:** A diagram showing a sequence of instructions with arrows indicating flow. Below it, the text "analyze (function)" is written.
- Middle Left:** A diagram showing a sequence of instructions with arrows indicating flow. Below it, the text "analyze (function)" is written.
- Middle Center:** A diagram showing a sequence of instructions with arrows indicating flow. Below it, the text "analyze (function)" is written.
- Middle Right:** A diagram showing a sequence of instructions with arrows indicating flow. Below it, the text "analyze (function)" is written.
- Bottom Left:** A diagram showing a sequence of instructions with arrows indicating flow. Below it, the text "analyze (function)" is written.
- Bottom Center:** A diagram showing a sequence of instructions with arrows indicating flow. Below it, the text "analyze (function)" is written.
- Bottom Right:** A diagram showing a sequence of instructions with arrows indicating flow. Below it, the text "analyze (function)" is written.

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



Step 8. Binary 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.