Decompiler

2016017 - Arnav Kumar | 2016095 - Shwetank Shrey | 2016268 - Siddharth Yadav

We submitted the ASM file as the input to the Code.

Firstly the python code separates the ASM file into the subroutines as they exist in the ASM. These are then added into an array.Once the subroutines are identified, they are added to an array, where they await individual processing to decompile the ASM. Now we do something that is kind of control graph making. Once individual subroutines are identified, we run an "identify loops and ifs" method to get the list of list of loops and ifs and if-elses. We then select the outermost loop/conditional block and convert the branch and compare statements to respective if/do-while statements along with proper braces. The code outside the outermost block is translated corresponding to the declarative or arithmetic or logical or interrupt statement and dealt with accordingly. The block of code inside the loop is then passed recursively to the "identify loops and ifs" method.
Identifiers and converters related to each statement are made. A list is created to check if the variables corresponding to the registers have been declared or not.

Detecting functions:

Every subroutine has input parameters as well as an output. The input parameters are identified using the default registers(s0, s1, s2, r0, r1, r2, d0, d1, d2 etc) used for passing parameters in ARM. We basically consider anything as a parameter if has been used in other places without initialization. The type of input parameter are figured out by using the kind of ldr, str and mov used along with it(example ldrb implies char, vstr.32 implies float). The number of parameters is also identified in the same manner. The start and end of subroutines is identified through pushes and pops with reference to the stack pointer.

Detecting loops:

Places where `cmp` is followed by `b<condition>` and the `label` of branching statement lies *above* the `cmp` is termed as loop.

Detecting ifs:

Places where `cmp` is followed by `b<condition>` and the `label` of branching statement lies *below* the `cmp` is termed as loop.

Detecting data types:

ARM uses different kinds of ldr, str, mov, add, sub, mul, div for different kind of data types.

For examples:
ldrb, strb implies use of char, vmov.f32 implies use of float, *vldr.64 implies double[not implemented due to use of adr in a weird way]*

Detecting SWI:
Places where swi is used, it is checked whether the next token is code for read, write or exit (0x6c, 0x6b or 0x11) then the decompiled code (scanf, printf or exit) is added to a string and the string is added to a list. The final contents of the list are then added to a file.

The ethics or the way of doing anything in arm code(for example the way to make or call a function) must be similar to that of arm-linux-gnueabihf-gcc.

We have a private git repo @ https://github.com/geekSiddharth/decompiler