Craig Green
EN 695.744 – Programming Assignment 1
cgree110@jh.edu

Linear Sweep & Recursive Descent Disassembly Algorithms

Disassembling binary to assembly code presents its challenges. Because of the dynamic nature of the program executing, and unknown values stored in memory, the disassembling process is at a disadvantage. This paper will discuss the two well known approaches to disassembly – linear sweep and recursive descent – as well as the strengths and weaknesses of both.

The linear sweep algorithm, as it is named, linearly travels through the given binary, byte by byte, to determine the instructions within. Since every instruction has a given opcode and instruction signature, the linear sweep algorithm can simply map the binary given to the net instruction. Additionally, any relative addressing can be calculated because the program counter is either given within the headers of the input file, or assumed to be zero.

This approach to disassembling is intellectually simple to comprehend, and because of that is simpler to implement. Additionally, because the algorithm steps through every byte, we can be sure that every byte has indeed been disassembled. Linear sweep begins to run into complications however when control-flow instructions are introduced. Because some control-flow instructions reference memory, branch addresses can only be known at runtime. This means that the program may not execute in a simple linear fashion that the algorithm is implying.

Recursive descent attempts to be more intelligent in regard to control-flow instructions. This algorithm traverses through the program in a linear fashion, but also considers the branch addresses within control-flow instructions as new starting points for disassembly. Control-flow instructions that reference immediate offsets rather than memory can be calculated at the time of disassembly. The program is first disassembled in a linear sweep fashion, keeping a reference of each target offset location that was calculated. Then the target offsets can be used a new starting points for further disassembly.

Recursive descent is inherently more complex to implement, since the control flow instructions will now modify the path that the disassembler will ultimately take. However, it is typically more accurate because of its ability to disassemble the code in the order that it will actually execute.

Craig Green
EN 695.744 – Programming Assignment 1
cgree110@jh.edu

IDA Pro is an effective, albeit not perfect, disassembler because it provides the engineer with the preferable recursive descent approach to disassembly, but also allows any individual instruction to be manually changed to be interpreted as a different type of instruction [1]. An engineer with experience can look at a block of assembly and determine that the disassembler is actually confused about a given code block, and tell it to reinterpret that block as either a certain instruction, or just data. IDA Pro will then re-disassemble that block.

Craig Green
EN 695.744 – Programming Assignment 1
cgree110@jh.edu

Sources

[1] Hex-Rays IDA Pro, https://hex-rays.com/ida-pro