Web Assembly Part - II Report

COP 701: Software Systems Lab

(MCS192568 and MCS192574)

Introduction

In this part of the assignment we will find the trace of a particular program and convert all the instructions of web assembly to equivalent x86 architecture instructions.

File Manual

1. **Log-all.js** - The code used for analysis in the WebAssembly front end. This code invokes ajax calls to the back-end python script and enables the translation of instructions and storing them in a zip file “on the fly”.
2. **extractBaseAddress.py** - This python script is used for extracting the memory addresses of each instruction from the wasm-objdump -d command. This python file generates a pickle file which is used in the main python file to implement program counter logic.
3. **mapX86.csv** - This csv file contains mapping of each web assembly instructions to equivalent x86 architecture instructions.
4. **PythonX86Mapper.py** - The main python script that calls emrun command and captures the instruction trace from web assembly and stores it in the gz file on the fly.
5. **Run.sh** - The final shell executable of the entire project. This shell script executes all of the above files and other commands and generates the output in the final gz file of the form of “<filename>.gz”

Implementation

The implementation of the required task is as follows:-

- An objdump pickle is generated using wasm-objdump command and extractBaseAddress.py script.
- We use subprocess.Popen to ensure block-wise receiving the instructions from the front-end. The block size considered by us is 1000 instructions. Emrun is used for running web assembly
- We read the mapX86.csv and map each instruction of web assembly to x86 format.
- We store the output is <filename>.gz file on the fly.

How to Run

The project can be run by issuing the following command:-

./run.sh <trace size> <path-to-program-name>

Now program name should contain path to the program in case it is not in the same folder as all the scripts.

If it is in the same folder, then pass “./<program-name>”

Now in the same folder an output.gz file is generated which contains the trace of required wasm file till required number of instructions.

Difficulties Faced

- Getting the javascript instruction trace on the fly
- The main difficulty was creating a map of web assembly instructions in corresponding x86 format.
- We got some instructions whose corresponding map in x86 wasn’t available. Hence to prevent loss of synchronization in the program counter value, we have considered those instructions as NOP instructions.

Additional Implementations:

- In Web Assembly binary and unary instructions like ADD don’t return the addresses but rather return the values to be added and the result of additions.
- So, we have Implemented Global Stack and Memory allocations, in which each function is allocated 1KB of stack and 3KB of memory size.
- Using this we are also able to virtualize the address of the operands