CS6383: Assignment 1
Finding the Footprint of a Variable
Due Saturday September 28th, 2019 at 11:59 PM

Introduction In this assignment, you are required to write an analysis pass in LLVM (version 9.0) to print the footprint of a given variable in high level program from its LLVM-IR module.

Footprint of a variable in high level language is the set of line numbers in source code where the given variable is being accessed.

```
1  int foo() {
2    int i = 0;
3    int *ip=&i;
4    for (int i = 0; i < 100; i++) {
5      int temp = 0;
6      temp += i;
7      *ip = temp;
8    }
9    return i;
10 }
```

Listing 1: Source Code

In the given example 1, variable i has two set of footprints as there are two different variables with same name and diffefrent scope. The variable i with scope (2:10) has footprint of (2,3,7,9) whereas the variable i with scope (4:8) has footprint of (4,6). In this assignment, you need to find the footprint for the given variable along with other information as specified below. You need to write an analysis pass that can be loaded dynamically and run from LLVM pass pipeline using *opt*.

Your LLVM pass must be generic enough to be scheduled at any position of the LLVM pass pipeline. Proper API must be provided so that the analysis information can be used by other LLVM passes.

Your pass should print the following information to the output stream. You must follow this format strictly as the testing will be done through scripts.

Clang version : <Version name>
LLVM Source Repository : <git/svn repository path>
LLVM Commit Hash : <commit hash>
Clang Source Repository : <git/svn repository path>
LLVM Commit Hash : <commit hash>
Target : <Target Triplet>

Variable Name : <Given Variable>
Variable Scope : <Line Numbers of Source Code.>
Footprint : <Comma separated line numbers>
Number of Reads : <Number of times the variable is read>
Number of Writes : <Number of times the variable is written to>
...

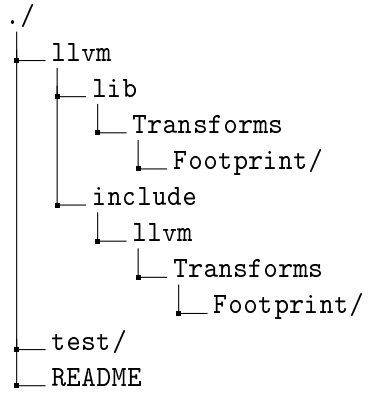
For your reference, the output for footprint of variable *i* in the given example 1 compiled with Clang version 8.0.1 on an Intel system running Linux should be as follows:
$ *opt -load $LLVM_ BUILD/lib/Footprint.so -ftprint -var-name=i test.ll*

Clang version : 8.0.1
LLVM Source Repository : https://git.llvm.org/git/llvm.git
LLVM Commit Hash : ea28a67e47fd87c6b78597d90eba543bad4d7468
Clang Source Repository : https://git.llvm.org/git/clang.git
LLVM Commit Hash : 2e4c9c5fc864c2c432e4c262a67c42d824b265c6
Target : x86_64-unknown-linux-gnu
Variable Name : i
Variable Scope : 2:10
Footprint : 2, 3, 7, 9
Number of Reads : 2
Number of Writes : 101
Variable Name : i
Variable Scope : 4:8
Footprint : 4, 6
Number of Reads : 301
Number of Writes : 101

Assumptions

- Input variable will always be part of source.
- IR can be in SSA as well as non-SSA form.
- Consider loops with unit step size. Handling loops with non-unit increment will fetch bonus points.
- If loop bounds are parametric, and the variable is accessed inside the loop, consider number of reads and writes as undefined and print it as *undef* in the output.

Implementation Guidelines Create a new directory *Footprint* in lib/Transforms/ folder of the LLVM source tree. All your code should be in this directory as this directory will be part of your submission. You need to put header files under include/llvm/Transforms/ directory. Register your pass as `ftprint` and the module as `Footprint.so` so that it can be run using scripts. Variable name (*var*) should be passed as command line argument `var-name`.
(ex. *opt -load $LLVM_ BUILD/lib/Footprint.so -ftprint -var-name=var test.ll* should work)
Strictly follow the below directory structure while submitting your code. Submit new and modified files only.



Testing You are supposed to write non-trivial test cases to test your pass. The test cases should vary in complexity from simple to more complex ones. You need to submit at least 5 test cases in C/C++.

Submission Your submission should be a bzip2 archive with name *Asgn1_ROLLNO.bzip2* containing the source code, header files, test-cases directory and a README file in the specified format. The README should mention all the materials that you have read/used for this assignment including LLVM documentation and source files. Mention the status of your submission, in case some part of it is incomplete. You can also include feedback, like what was challenging and what was trivial. Include files that you have added or modified. Do not include the binaries in your submission.

Evaluation We will test your code using a set of 50 test cases and you will be evaluated based on the number of test cases passed. Also proper commenting, code formatting along with well structured code will be part of the evaluation and fetch you more points.
NOTE: Non-compliance to the submission guidelines will attract strict penalty.