CS6383: Assignment 1
Source Statement mapping from LLVM IR
Due Wednesday September 19th, 2018 at 11:59 PM

Introduction In this assignment, you will write an LLVM pass to map list of LLVM IR instructions to corresponding high level statements involving **arithmetic operations**.

```
1 int main(){
2    int A[100], B[100], C[100];
3    //init A, B, C
4    for (int i=0; i<100; i++)
5       C[i] = A[i] + B[i]; //S1
6    return 0;
7 }
```

Listing 1: Source Code

```
%2 = alloca [100 x i32], align 16
%3 = alloca [100 x i32], align 16
%4 = alloca [100 x i32], align 16
%6 = alloca i32, align 4
%29 = load i32, i32* %6, align 4
%30 = sext i32 %29 to i64
%31 = getelementptr inbounds [100 x i32], [100 x i32]* %2, ↩
    i64 0, i64 %30
%32 = load i32, i32* %31, align 4
%33 = load i32, i32* %6, align 4
%34 = sext i32 %33 to i64
%35 = getelementptr inbounds [100 x i32], [100 x i32]* %3, ↩
    i64 0, i64 %34
%36 = load i32, i32* %35, align 4
%37 = add nsw i32 %32, %36
%38 = load i32, i32* %6, align 4
%39 = sext i32 %38 to i64
%40 = getelementptr inbounds [100 x i32], [100 x i32]* %4, ↩
    i64 0, i64 %39
store i32 %37, i32* %40, align 4
```

Listing 2: LLVM IR

The statement $S1$ at line number 5 in Listing 1 corresponds to all the LLVM IR instructions showed in Listing 2. For this example, the output of your pass should be Statement=5{<all the instructions from Listing 2>}.

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.

Statement=<source code line number>{
<list of LLVM IR instructions>
}
...

Implementation Guidelines Create a new directory *StatementMapper* 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 `stmt-mapper` and the module as `StatementMapper.so` so that it can be run using scripts.
(ex. *opt -load $LLVM_BUILD/lib/StatementMapper.so -stmt-mapper test.ll* should work)
Strictly follow the below directory structure while submitting your code. Submit new and modified files only.

```
./
├── llvm
│   ├── lib
│   │   └── Transforms
│   │       └── StatementMapper/
│   └── include
│       └── llvm
│           └── Transforms
│               └── StatementMapper/
├── test/
└── README
```

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. 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.