CS6383: Assignment 2
Loop Unrolling in LLVM
Due Friday October 5th, 2018 at 11:59 PM

Goal In this assignment, you are required to write a transformation pass in LLVM, to unroll all the innermost loops in a function with a user specified unroll factor.

Introduction Loop Unrolling is a loop optimization technique where the number of loop iterations are reduced by increamenting the loop with an unroll factor and replicating the statements inside the loop. This transformation reduces the number of branches at the cost of code size. In many circumstances, loop unrolling enables other optimizations like vectorization and some peephole optimizations.

For an example, the sample loop in listing 1 can be unrolled with Unroll Factor (UF) as shown in listing 2.

```
1   for (int i=0; i < N; i++){
2     S1;
3     S2;
4   }
```

Listing 1: Sample Loop

```
1   for ( i=0; i < N/UF; i += UF){
2     S1;
3     S2;
4     S1';
5     S2';
6     //Repeat UF times
7   }
8   //epilogue
9   for ( ; i < N; i++){
10    S1;
11    S2;
12  }
```

Listing 2: Unrolled Sample Loop

A more specific example is shown below where the given loop in listing 3 is unrolled with $UF = 4$ as shown in listing 4. You are required to unroll all the innermost loops present in the source code.

Assumptions

- Handle innermost *for* loops only.
- 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.
- Do not unroll if UF is greater than the loop trip count, which will be a compile time constant.

```c
int main(){
   int A[100], B[100], C[100];
   //init A, B, C
   for (int i=0; i<100; i++)
      C[i] = A[i] + B[i];
   return 0;
}
```

Listing 3: Source Code

```c
int main(){
   int A[100], B[100], C[100];
   //init A, B, C
   for (int i=0; i<25; i+=4){
      C[i] = A[i] + B[i];
      C[i+1] = A[i+1] + B[i+1];
      C[i+2] = A[i+2] + B[i+2];
      C[i+3] = A[i+3] + B[i+3];
   }
   //epilogue is empty here
   return 0;
}
```

Listing 4: Unrolled with factor of 4

Your LLVM pass must be generic enough to be scheduled at any position of the LLVM pass pipeline.

Implementation Guidelines Create a new directory *LoopUnrolling* in lib/Transforms/ folder of the LLVM source tree. All your implementations 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 `loop-unrolling` and the module as `LoopUnroller.so` so that it can be run using scripts. Unroll Factor (UF) should be passed as command line argument `unroll-factor` with 4 as default UF.
(ex. *opt -load $LLVM_BUILD/lib/LoopUnroller.so -loop-unrolling -unroll-factor=UF test.ll* should work)
Strictly follow the below directory structure while submitting your code. Submit new and modified files only.

```
./
├── llvm
│   ├── lib
│   │   └── Transforms
│   │       └── LoopUnrolling/
│   └── include
│       └── llvm
│           └── Transforms
│               └── LoopUnrolling/
├── 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 *Asgn2_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.