CS536 Advanced Compiler

Assignment III (Instrument the code to generate LD/ST Memory Address Trace),

Deadline : 23rd October 2022 Thursday 11.59PM

(System will not accept submission after 24th October 11.59PM)

- Submission Procedure: Upload your code (zipped folder) to MS team. The assignment need to be done individually. **Copy case lead to F Grade**
- You need to demonstrate your code in your machine and TA will evaluate the same.

Create a LLVM pass, which take LLVM IR file of any C code and produce an instrumented IR file.

- For each load/store (LD/ST statement), add extra (instrumentation) call printL/printS instruction after the load/store instruction, which print the address.
- Address of LD/ST can be passed to printL/printS function as a parameter or global variable.
- You can append code for GobalAddr, printL and printS for beginning of input C code before generating the IR file. During the analysis pass you skip these two functions printL and printS

In this assignment, we need to modify the input program IR file, we not only read the IR file, but also modify the IR file to create new IR file. Your new IR file should compile and run to generate the address trace.

Running LLVM Pass means: $opt -passes=your_llvm_pass test.ll

Hint: You can get information by reading the CPP files of sub-directories of llvm/lib/Transforms.

Code for printL/printLL and printS/printSS:

//suppose this code is supportingcode.c

```
unsigned int GA;
FILE *mfp;
int firsttimeOpen=0;
void printL(){
//void printLL(unsigned int paramA){
      if (firsttimeOpen==0) {
            mfp=fopen(“mytrace.txt”,”w”);
            firsttimeOpen=1;
      }
      fprintf(mfp,“\n%u”,GA);
      //printf(mfp,“\n%u”,paramA);
}
void printS(){
//void printSS(unsigned int paramA){
      if (firsttimeOpen==0) {
            mfp=fopen(“mytrace.txt”,”w”);
            firsttimeOpen=1;
      }
      fprintf(mfp,“\n%u”,GA);
      //fprintf(mfp,“\n%u”,paramA);
}
```

```
$cat supportingcode.c testApp.c testAppWSC.c
$clang -emit-llvm -S testAppWSC.c -o testAppWSC.ll
```

Use this testAppWSC.ll as input for your transformation. You can use the same lbm.c as input or fft.c (uploaded to MS team).

For LD/ST instructions.

LD/ST R1 R3 //assuming R3 is address and operation R1 ←-M[R3] or M[R3]<--R1, Add two instructions after the LD/ST instructions

- Either add two instructions (a) GA=R3 and (b) call printL/printS
- or add two instructions (a) param1=R3 and (b) call printLL/printSS