**HelloWorld.cpp is in llvm-tutor-main\HelloWorld**

**The test files and the shell script files are in llvm-tutor-main\build**

**Date**: 1/30/2024

**Objective**: Install llvm-tutor-main, install test files, and run demo from TA's slides

**Tasks Completed**: Install llvm-tutor-main and test files

**Challenges** **Faced**:

1. ./test.sh didn't work in WSL.

**Solutions/Workarounds**:

1. dos2unix test.sh (command for wsl, shell file to work)

**Code Snippets**:

export LLVM\_DIR=/lib/llvm-17/

cd build

cmake -DLT\_LLVM\_INSTALL\_DIR=$LLVM\_DIR ..

make

clang -c -emit-llvm -fno-discard-value-names -O0 demo.cpp -o demo.bc

opt -load-pass-plugin ./lib/libHelloWorld.so -passes=hello-world -disable-output demo.bc

./demo.sh

**Testing**: Ran demo file to see function names and its arguments

**Learnings**: Introduced to HelloWorld's visitor function.

**Next Steps**: Finish project

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Date**: 2/2/2024

**Objective**: Finish project

**Tasks Completed**: Finished project

**Challenges** **Faced**:

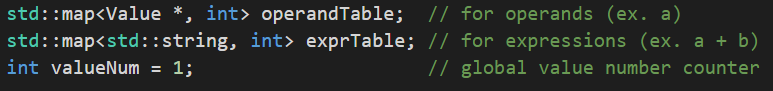
1. Issue with hashTable. Originally I had only one hashtable for both operands and expressions. However, there was conflict between the Value\* addresses for operands and the string expressions.
2. There was confusion on memory address occurrences for load and store in relation to what value number they should have.
3. Given how I implemented updateOperandTable, there were multiple instances of the same key with different value numbers, when I really wanted to update the previous value stored there.
4. I had a condition to check if the expression already existed in the table in the visitor function. It was strange extraneous code.

**Solutions/Workarounds**:

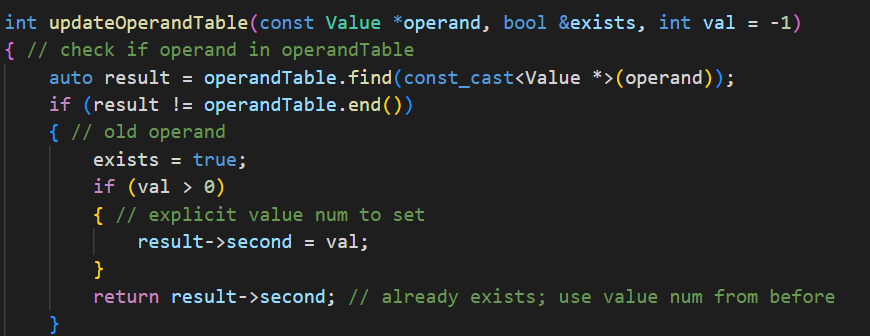
1. Instead of just one hashTable, I made an operandTable<Value\*,int> and exprTable<string,int>. Both tables share the same global value number.
2. I printed out addresses for load &inst and inst.getOperand(0) and store inst.getOperand(0) and inst.getOperand(1). Then I was able to see that placing inst.getOperand(0) in the operandTable for both load and store before placing their respective destination addresses in the operandTable.
3. I added a parameter "val" in updateOperandTable to allow specific assignment of a value number to the operand passed in. This removed the uncertainty of multiple value numbering mapped to a single operand.
4. In the end, I simplified everything by adding a parameter "exists" to update and determine if the key exists already in exprTable.

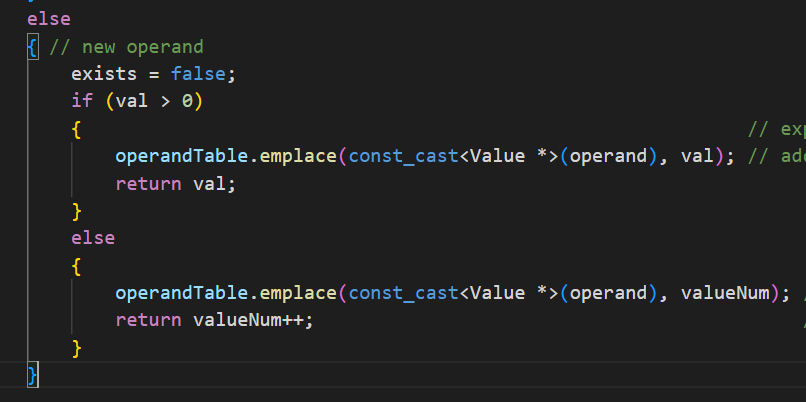
**Code Snippets**:

Definition of hash tables and global value number

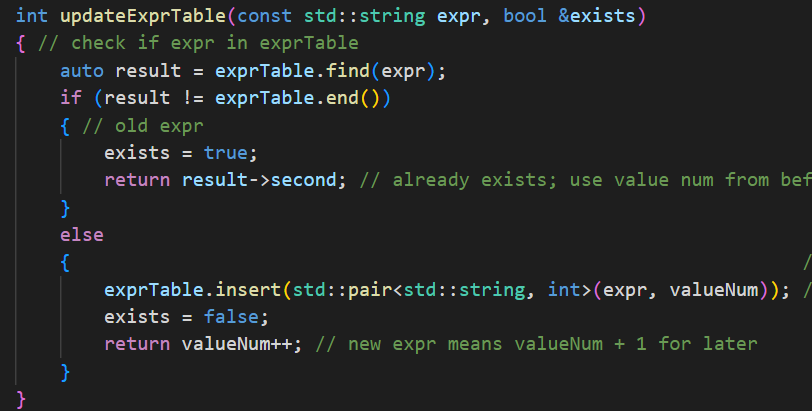


Function to update operandTable with optional option to specify value number with val

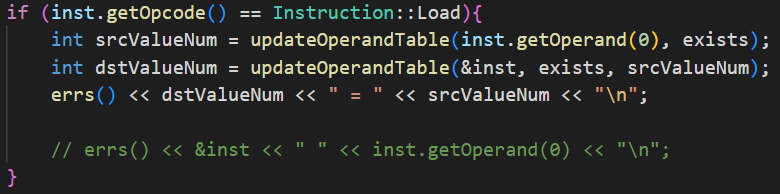




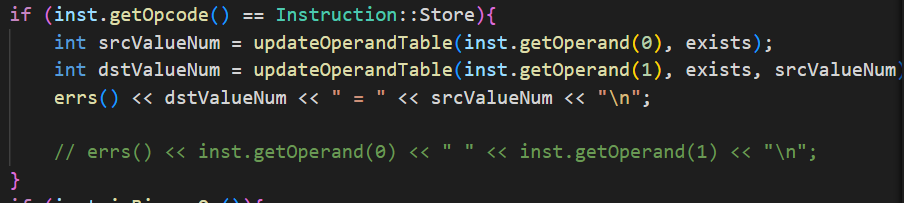
Function to update exprTable with the same global value number as operandTable.



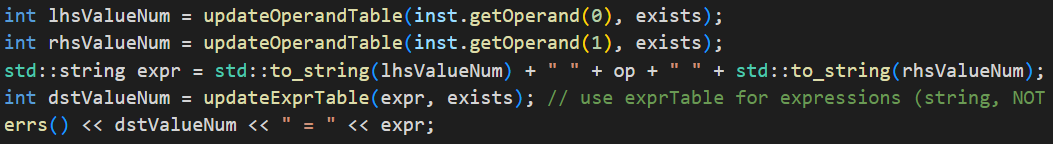
visitor() Load handler, place source operand into operandTable then destination with same value number

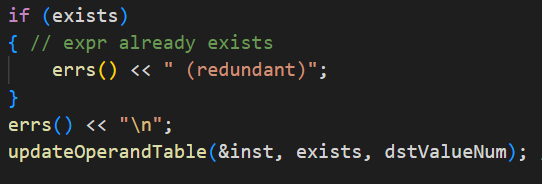


visitor() Store handler, place source operand into operandTable then destination with same value number



visitor() Expression handling, place expr into exprTable. If it's already there, state that the current expression is redundant.





**Testing**: Ran ./test.sh and the output matches exactly as given in the project description. Also created testDiv.c using the "/" operator.

**Learnings**: Learned about llvm passes and how to handle instructions in basic blocks in a function. I saw how particular instructions are structured and how to access their operands, and particularly how these address values relate to other instruction types (load and store). I learned about the process of value numbering and how the addresses of type Value\* should be stored and what value numbers particular addresses should point to.

**Next Steps**: Finish project