Project Report
Topic: Available expressions and Common Subexpression Elimination

Methodology:

To implement this Optimization on llvm code:

Analysis Pass:

I have implemented Analysis Pass-Available expressions using the algorithm (source:Unit 6 Support Material)

```
OUT[$Bexit$] = ∅
    for each basic block $B$ other than $Bexit$:
        OUT[$B$] = U (universal set with all expressions)
        while any OUT set has changes:
            for each basic block $B$:
                IN[$B$] = $\bigcap_{BP \in pred(B)}$ OUT[$BP$]
                OUT[$B$] = $genB$ ∪ (IN[$B$] − $kill\ B$)
```

Transform Pass:

After implementing the analysis pass of available expressions, I implemented the Transform pass Common sub-expression elimination, that replaces the instruction with the instruction that has the expression available. This is done by comparing the gen-set(expression) of the current instruction and in-set(expressions) of the current instruction. if the gen-set(expression) of current instruction is same as any inset of the current instruction, then the current instruction is replaced with the instruction that is providing the available expression in the in-set.

Testing:

For the implementation I created two passes.

- Available expressions with flag -avexp
- Common Subexpression Elimination with flag -cse

For testing the correctness of the implemented code, two test files are created:

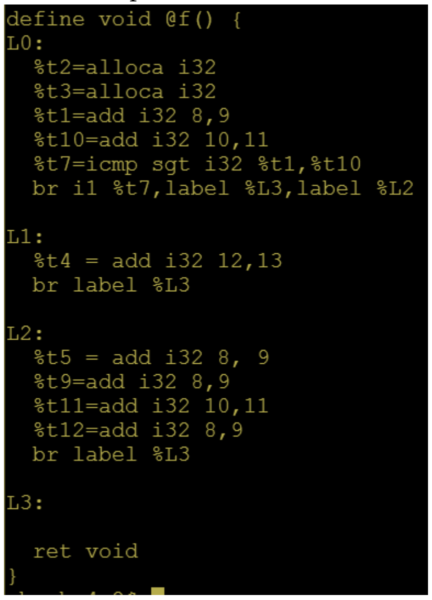
Now using the following command a *"pass.so"* is created to pass through the llvm code:

g++ -g -fPIC -shared available-exp.cc -o pass.so -std=c++11 `llvm-config --cppflags`

two llvm codes are passed with the following commands:

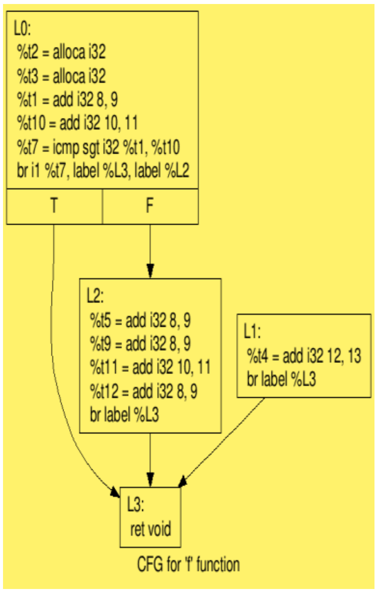
cat test1.ll | opt -mem2reg -load ./pass.so -avexp -cse(Multiple Basic Blocks)

Input Code



```
define void @f() {
L0:
  %t2=alloca i32
  %t3=alloca i32
  %t1=add i32 8,9
  %t10=add i32 10,11
  %t7=icmp sgt i32 %t1,%t10
  br i1 %t7,label %L3,label %L2

L1:
  %t4 = add i32 12,13
  br label %L3

L2:
  %t5 = add i32 8, 9
  %t9=add i32 8,9
  %t11=add i32 10,11
  %t12=add i32 8,9
  br label %L3

L3:

  ret void
}
```

CFG



```
L0:
%t2 = alloca i32
%t3 = alloca i32
%t1 = add i32 8, 9
%t10 = add i32 10, 11
%t7 = icmp sgt i32 %t1, %t10
br i1 %t7, label %L3, label %L2
```

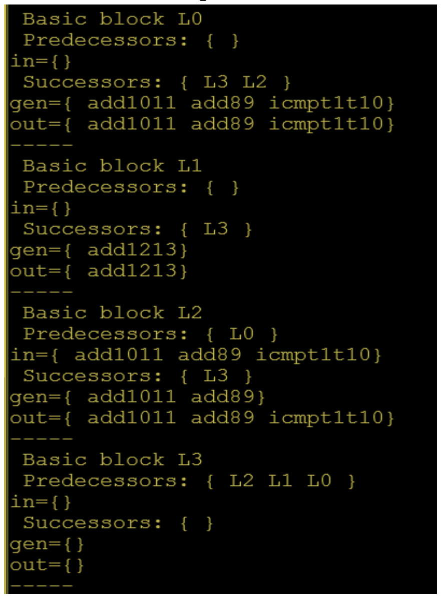
```
L2:
%t5 = add i32 8, 9
%t9 = add i32 8, 9
%t11 = add i32 10, 11
%t12 = add i32 8, 9
br label %L3
```

```
L1:
%t4 = add i32 12, 13
br label %L3
```

```
L3:
ret void
```

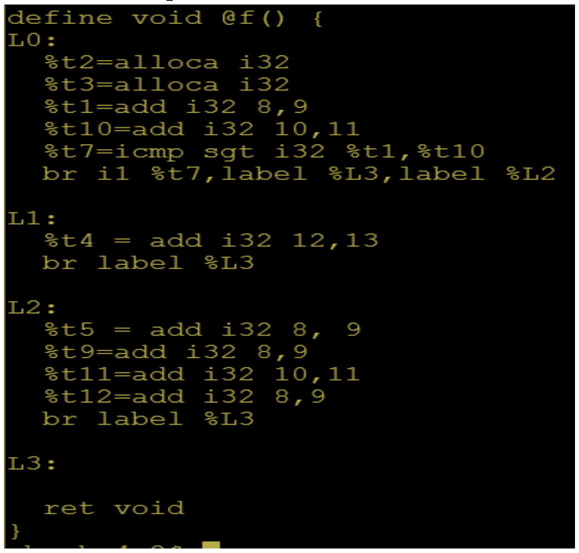
CFG for 'f' function

Available Expressions

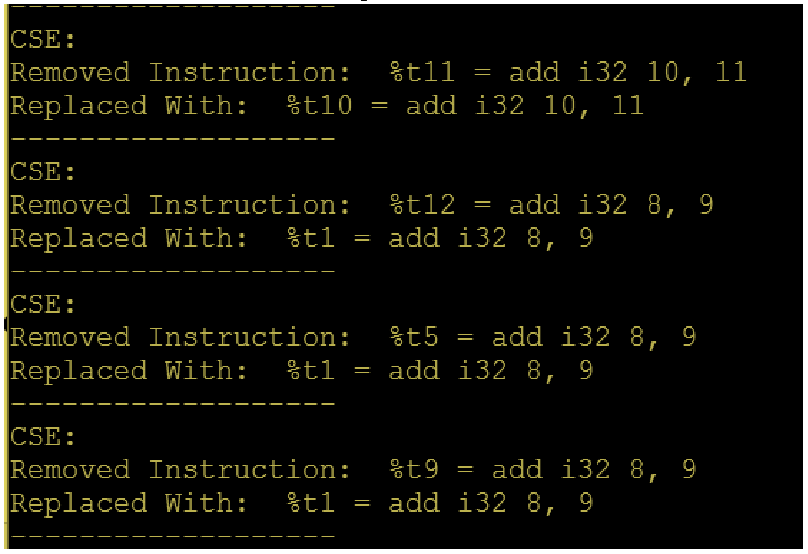


```
 Basic block L0
 Predecessors: { }
in={}
 Successors: { L3 L2 }
gen={ add1011 add89 icmpt1t10}
out={ add1011 add89 icmpt1t10}
-----
 Basic block L1
 Predecessors: { }
in={}
 Successors: { L3 }
gen={ add1213}
out={ add1213}
-----
 Basic block L2
 Predecessors: { L0 }
in={ add1011 add89 icmpt1t10}
 Successors: { L3 }
gen={ add1011 add89}
out={ add1011 add89 icmpt1t10}
-----
 Basic block L3
 Predecessors: { L2 L1 L0 }
in={}
 Successors: { }
gen={}
out={}
-----
```

Input Code *Common Sub-Expression-Elimination*



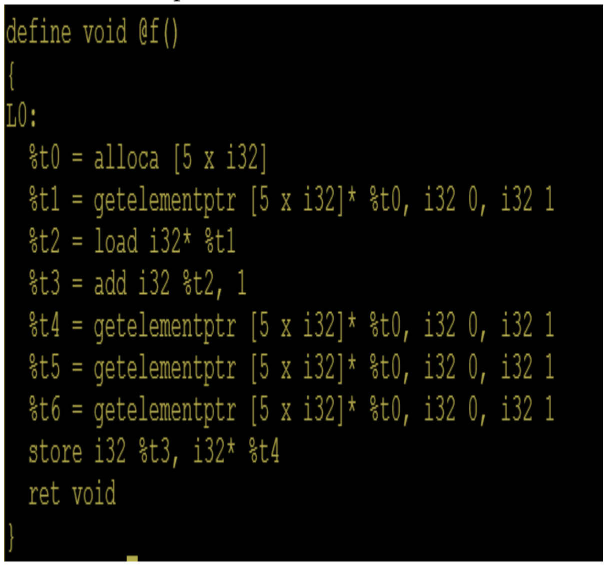
```
define void @f() {
L0:
  %t2=alloca i32
  %t3=alloca i32
  %t1=add i32 8,9
  %t10=add i32 10,11
  %t7=icmp sgt i32 %t1,%t10
  br i1 %t7,label %L3,label %L2

L1:
  %t4 = add i32 12,13
  br label %L3

L2:
  %t5 = add i32 8, 9
  %t9=add i32 8,9
  %t11=add i32 10,11
  %t12=add i32 8,9
  br label %L3

L3:

  ret void
}
```



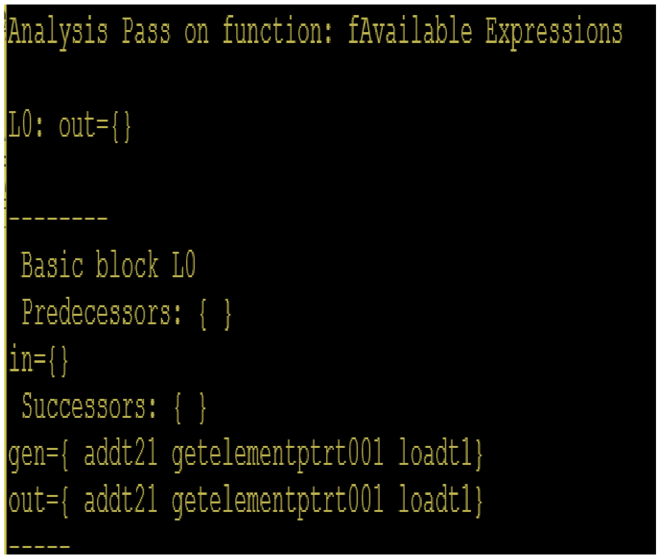
```
CSE:
Removed Instruction:  %t11 = add i32 10, 11
Replaced With:  %t10 = add i32 10, 11
--------------------
CSE:
Removed Instruction:  %t12 = add i32 8, 9
Replaced With:  %t1 = add i32 8, 9
--------------------
CSE:
Removed Instruction:  %t5 = add i32 8, 9
Replaced With:  %t1 = add i32 8, 9
--------------------
CSE:
Removed Instruction:  %t9 = add i32 8, 9
Replaced With:  %t1 = add i32 8, 9
--------------------
```

cat test2.ll | opt -mem2reg -load ./pass.so -avexp -cse(Single Basic Block)

Input Code *Available Expressions*

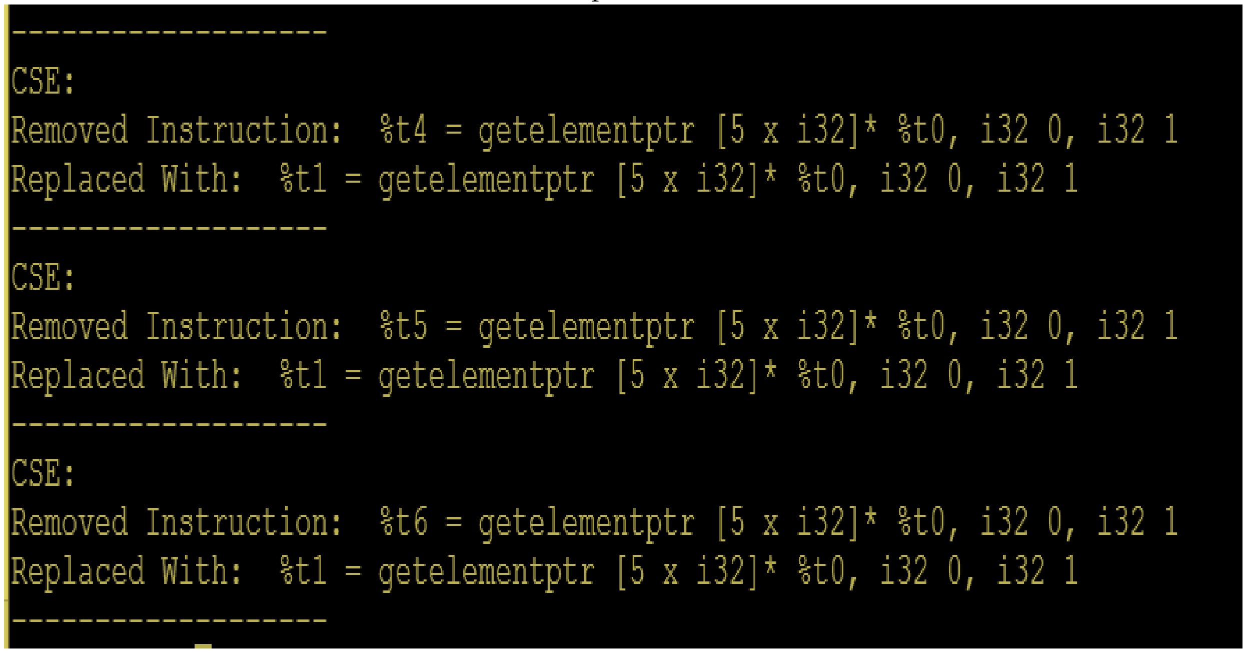


```
define void @f()
{
L0:
  %t0 = alloca [5 x i32]
  %t1 = getelementptr [5 x i32]* %t0, i32 0, i32 1
  %t2 = load i32* %t1
  %t3 = add i32 %t2, 1
  %t4 = getelementptr [5 x i32]* %t0, i32 0, i32 1
  %t5 = getelementptr [5 x i32]* %t0, i32 0, i32 1
  %t6 = getelementptr [5 x i32]* %t0, i32 0, i32 1
  store i32 %t3, i32* %t4
  ret void
}
```



```
Analysis Pass on function: fAvailable Expressions

L0: out={}


--------
 Basic block L0
 Predecessors: { }
in={}
 Successors: { }
gen={ addt21 getelementptrt001 loadt1}
out={ addt21 getelementptrt001 loadt1}
-----
```

Common Subexpression Elimination:



```
--------------------
CSE:
Removed Instruction:  %t4 = getelementptr [5 x i32]* %t0, i32 0, i32 1
Replaced With:  %t1 = getelementptr [5 x i32]* %t0, i32 0, i32 1
--------------------
CSE:
Removed Instruction:  %t5 = getelementptr [5 x i32]* %t0, i32 0, i32 1
Replaced With:  %t1 = getelementptr [5 x i32]* %t0, i32 0, i32 1
--------------------
CSE:
Removed Instruction:  %t6 = getelementptr [5 x i32]* %t0, i32 0, i32 1
Replaced With:  %t1 = getelementptr [5 x i32]* %t0, i32 0, i32 1
--------------------
```