DCSE, CEG, ANNA UNIVERSITY CHENNAI - 600025

ME – CSE

CP5252 – COMPILER OPTIMIZATION TECHNIQUES

LABORATORY RECORD

Reg No - 2021207032
Name – S. SUSHMITHA
Cell No - 6383176758
Email – sushmithaselvam1994@gmail.com

INSTRUCTOR: **Dr R. AROCKIA XAVIER ANNI**

INDEX

EXERCISE 1

Exercise 1: **INTRODUCTION TO LLVM AND INSTALATION**

AIM:

To learn about LLVM and to install it

LLVM:

- LLVM is a set of compiler and toolchain technologies, that can be used to develop afront end for any programming language and a back end for any instruction set architecture.
- LLVM is designed around a language independent intermediate representation (IR) thatserves as a portable, high-level assembly language that can be optimised with a varietyof transformations over multiple passes.
- LLVM can provide the middle layers of a complete compiler system, taking intermediate representation (IR) code from a compiler and emitting an optimised IR. This new IR canthen be converted and linked into machine-dependent assembly language code for a target platform.
- LLVM supports a language-independent instruction set and type system.
- Each instruction is in static single assignment form (SSA), meaning that each variable (called a typed register) is assigned once and then frozen.
- This helps simplify the analysis of dependencies among variables.
- LLVM allows code to be compiled statically, as it is under the traditional GCC system, or left for late-compiling from the IR to machine code via just-in-time compilation (JIT), similar to Java

STEP TO INSTALL LLVM:

- Open Terminal
- Enter command $ sudo apt install LLVM

RESULT:

Thus LLVM installation has been implemented

Exercise 2: **CLANG INSTALLATION AND AN EXAMPLE**

AIM:

To install Clang and to implement an example

CLANG

- Clang is a compiler front end for the C, C++, Objective-C, and Objective-C++ programming languages, as well as the OpenMP, OpenCL, RenderScript, CUDA,and HIP frameworks.
- It acts as a drop-in replacement for the GNU Compiler Collection (GCC), supportingmost of its compilation flags and unofficial language extensions.
- It includes a static analyzer, and several code analysis tool
- Clang works in tandem with LLVM

STEPS TO INSTALL:

- Open terminal
- Run command $ sudo apt install Clang

EXAMPLE USING LLVM TOOLCHAIN:

1. **CREATE A SIMPLE C FILE**

```
#include<stdio.
h>int main()
{
  printf("HELLO WORLD");
  Return 0;
}
```

2. **COMPILE THE C FILE TO NATIVE EXECUTABLE**

```
$ clang hello.c -ohello
```

3. **COMPILE THE C FILE INTO AN LLVM BIT CODEFILE**

```
$Clang -03 -emit-llvm hello.c -c -o hello.bc
```

4. **WE CAN RUN THE BITCODE FILE LLI(LOW LEVEL INTERPRETER)**

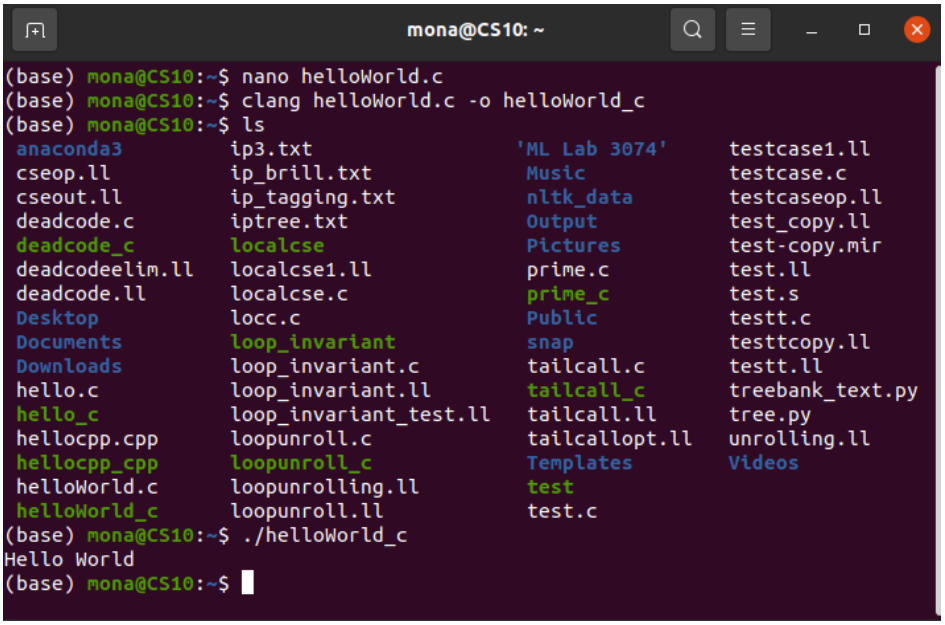
```
$ lli hello.bc
```

5. **COMPILE THE BITCODE INTO NATIVE ASSEMBLY CODE USING LLC**

$ llc hello.bc -o hello.s

OUTPUT:

C FILE



RESULT :

Thus CLANG has been installed and an example program has been implemented

Exercise 3: **COPY PROPAGATION**

AIM:

To implement copy propagation using LLVM.

CODING

```
#include<stdio.h>
int main()
{
int a;
scanf("%d",&a);
int b=a;
int c=b+4;
printf(" c value after copy propagation %d\n",c);
return 0;
}
```

1)clang -emit-llvm -S copy.c -o copy.ll -Xclang -disable-O0-optnone
2)opt -mem2reg copy.ll -S -o copy-test.ll

copy.ll

```
; ModuleID = 'copy.c'
source_filename = "copy.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
@.str.1 = private unnamed_addr constant [36 x i8] c" c value after copy
propagation %d\0A\00", align 1
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  %4 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  %5 = call i32 (i8*, ...) @__isoc99_scanf(i8* noundef getelementptr inbounds ([3
x i8], [3 x i8]* @.str, i64 0, i64 0), i32* noundef %2)
  %6 = load i32, i32* %2, align 4
```

```
  store i32 %6, i32* %3, align 4
  %7 = load i32, i32* %3, align 4
  %8 = add nsw i32 %7, 4
  store i32 %8, i32* %4, align 4
  %9 = load i32, i32* %4, align 4
  %10 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([36 x i8],
[36 x i8]* @.str.1, i64 0, i64 0), i32 noundef %9)
  ret i32 0
}
declare i32 @__isoc99_scanf(i8* noundef, ...) #1
declare i32 @printf(i8* noundef, ...) #1
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic" }
attributes #1 = { "frame-pointer"="all" "no-trapping-math"="true" "stack-
protector-buffer-size"="8" "target-cpu"="x86-64" "target-
features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
```

copy-test.ll

```
; ModuleID = 'copy.ll'
source_filename = "copy.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
@.str.1 = private unnamed_addr constant [36 x i8] c" c value after copy
propagation %d\0A\00", align 1
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
```

```
  %2 = call i32 (i8*, ...) @__isoc99_scanf(i8* noundef getelementptr inbounds ([3
x i8], [3 x i8]* @.str, i64 0, i64 0), i32* noundef %1)
  %3 = load i32, i32* %1, align 4
  %4 = add nsw i32 %3, 4
  %5 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([36 x i8],
[36 x i8]* @.str.1, i64 0, i64 0), i32 noundef %4)
  ret i32 0
}
declare i32 @__isoc99_scanf(i8* noundef, ...) #1
declare i32 @printf(i8* noundef, ...) #1
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic" }
attributes #1 = { "frame-pointer"="all" "no-trapping-math"="true" "stack-
protector-buffer-size"="8" "target-cpu"="x86-64" "target-
features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
```

RESULT :

Thus copy propagation has been implemented successfully.

Exercise 4: **CONSTANT PROPAGATION**

AIM:

To implement constant propagation using LLVM.

CODING

```
#include<stdio.h>
int main()
{
int a,b,c;
a=30;
b=20-a/2;
c=b*(30/a+2);
printf("%d",c);
return 0;
}
```

1)clang -emit-llvm -S constant.c -o constant.ll -Xclang -disable-O0-optnone
2)opt -mem2reg constant.ll -S -o constant-test.ll

constant.ll

```
; ModuleID = 'constant.c'
source_filename = "constant.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  %4 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  store i32 30, i32* %2, align 4
  %5 = load i32, i32* %2, align 4
  %6 = sdiv i32 %5, 2
  %7 = sub nsw i32 20, %6
  store i32 %7, i32* %3, align 4
  %8 = load i32, i32* %3, align 4
```

```
  %9 = load i32, i32* %2, align 4
  %10 = sdiv i32 30, %9
  %11 = add nsw i32 %10, 2
  %12 = mul nsw i32 %8, %11
  store i32 %12, i32* %4, align 4
  %13 = load i32, i32* %4, align 4
  %14 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([3 x i8], [3
x i8]* @.str, i64 0, i64 0), i32 noundef %13)
  ret i32 0
}
declare i32 @printf(i8* noundef, ...) #1
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic" }
attributes #1 = { "frame-pointer"="all" "no-trapping-math"="true" "stack-
protector-buffer-size"="8" "target-cpu"="x86-64" "target-
features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
```

constant-test.ll

```
; ModuleID = 'constant.ll'
source_filename = "constant.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = sdiv i32 30, 2
  %2 = sub nsw i32 20, %1
  %3 = sdiv i32 30, 30
  %4 = add nsw i32 %3, 2
```

```
  %5 = mul nsw i32 %2, %4
  %6 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([3 x i8], [3
x i8]* @.str, i64 0, i64 0), i32 noundef %5)
  ret i32 0
}
declare i32 @printf(i8* noundef, ...) #1
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic" }
attributes #1 = { "frame-pointer"="all" "no-trapping-math"="true" "stack-
protector-buffer-size"="8" "target-cpu"="x86-64" "target-
features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
```

RESULT :

Thus constant propagation has been implemented successfully.

Exercise 5: COMMON SUBEXPRESSION ELIMINATION

AIM:

To implement common subexpression elimination using LLVM.

CODING

```
#include<stdio.h>
int main()
{
int a=20,b=30,c,d,e;
c=a+b;
d=c*50;
e=(a+b)*50;
return 0;
}
```

1)clang -emit-llvm -S cse.c -o cse.ll -Xclang -disable-O0-optnone
2)opt -mem2reg cse.ll -S -o cse-test.ll

cse.ll

```
; ModuleID = 'cse.c'
source_filename = "cse.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  %4 = alloca i32, align 4
  %5 = alloca i32, align 4
  %6 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  store i32 20, i32* %2, align 4
  store i32 30, i32* %3, align 4
  %7 = load i32, i32* %2, align 4
  %8 = load i32, i32* %3, align 4
  %9 = add nsw i32 %7, %8
  store i32 %9, i32* %4, align 4
```

```
  %10 = load i32, i32* %4, align 4
  %11 = mul nsw i32 %10, 50
  store i32 %11, i32* %5, align 4
  %12 = load i32, i32* %2, align 4
  %13 = load i32, i32* %3, align 4
  %14 = add nsw i32 %12, %13
  %15 = mul nsw i32 %14, 50
  store i32 %15, i32* %6, align 4
  ret i32 0
}
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic" }
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
```

cse-test.ll

```
; ModuleID = 'cse.ll'
source_filename = "cse.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = add nsw i32 20, 30
  %2 = mul nsw i32 %1, 50
  %3 = add nsw i32 20, 30
  %4 = mul nsw i32 %3, 50
  ret i32 0
}
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
```

```
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic" }

!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
```

RESULT :

Thus common subexpression elimination has been implemented successfully.

Exercise 6: **LOOP INVARIANT CODE MOTION**

AIM:

To implement loop invariant code motion using LLVM.

CODING

```
#include<stdio.h>
int main()
{
int a;
int test=0;
int sum=0;
scanf("%d",&a);
for(int i=0;i<100;i++)
{
test=a*10;
sum=sum+i;
}
printf("%d\t%d\n",test,sum);
return 0;
}
```

1)clang -emit-llvm -S licm.c -o licm.ll -Xclang -disable-O0-optnone
2)opt -mem2reg licm.ll -S -o licm-test.ll

licm.ll

```
; ModuleID = 'licm.c'
source_filename = "licm.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
@.str.1 = private unnamed_addr constant [7 x i8] c"%d\09%d\0A\00", align 1
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  %4 = alloca i32, align 4
  %5 = alloca i32, align 4
```

```
  store i32 0, i32* %1, align 4
  store i32 0, i32* %3, align 4
  store i32 0, i32* %4, align 4
  %6 = call i32 (i8*, ...) @__isoc99_scanf(i8* noundef getelementptr inbounds ([3
x i8], [3 x i8]* @.str, i64 0, i64 0), i32* noundef %2)
  store i32 0, i32* %5, align 4
  br label %7
7:                                ; preds = %16, %0
  %8 = load i32, i32* %5, align 4
  %9 = icmp slt i32 %8, 100
  br i1 %9, label %10, label %19
10:                               ; preds = %7
  %11 = load i32, i32* %2, align 4
  %12 = mul nsw i32 %11, 10
  store i32 %12, i32* %3, align 4
  %13 = load i32, i32* %4, align 4
  %14 = load i32, i32* %5, align 4
  %15 = add nsw i32 %13, %14
  store i32 %15, i32* %4, align 4
  br label %16
16:                               ; preds = %10
  %17 = load i32, i32* %5, align 4
  %18 = add nsw i32 %17, 1
  store i32 %18, i32* %5, align 4
  br label %7, !llvm.loop !6
19:                               ; preds = %7
  %20 = load i32, i32* %3, align 4
  %21 = load i32, i32* %4, align 4
  %22 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([7 x i8], [7
x i8]* @.str.1, i64 0, i64 0), i32 noundef %20, i32 noundef %21)
  ret i32 0
}
```

licm-test.ll

```
; ModuleID = 'licm.ll'
source_filename = "licm.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
```

```
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
@.str.1 = private unnamed_addr constant [7 x i8] c"%d\09%d\0A\00", align 1
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = call i32 (i8*, ...) @__isoc99_scanf(i8* noundef getelementptr inbounds ([3
x i8], [3 x i8]* @.str, i64 0, i64 0), i32* noundef %1)
  br label %3
3:                                                ; preds = %9, %0
  %.02 = phi i32 [ 0, %0 ], [ %8, %9 ]
  %.01 = phi i32 [ 0, %0 ], [ %7, %9 ]
  %.0 = phi i32 [ 0, %0 ], [ %10, %9 ]
  %4 = icmp slt i32 %.0, 100
  br i1 %4, label %5, label %11
5:                                                ; preds = %3
  %6 = load i32, i32* %1, align 4
  %7 = mul nsw i32 %6, 10
  %8 = add nsw i32 %.02, %.0
  br label %9
9:                                                ; preds = %5
  %10 = add nsw i32 %.0, 1
  br label %3, !llvm.loop !6
11:                                               ; preds = %3
  %12 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([7 x i8], [7
x i8]* @.str.1, i64 0, i64 0), i32 noundef %.01, i32 noundef %.02)
  ret i32 0
}
declare i32 @__isoc99_scanf(i8* noundef, ...) #1
declare i32 @printf(i8* noundef, ...) #1
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic" }
attributes #1 = { "frame-pointer"="all" "no-trapping-math"="true" "stack-
protector-buffer-size"="8" "target-cpu"="x86-64" "target-
features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}

!0 = !{i32 1, !"wchar_size", i32 4}
```

```
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
!6 = distinct !{!6, !7}
!7 = !{!"llvm.loop.mustprogress"}
```

RESULT :

Thus loop invariant code motion has been implemented successfully.

Exercise 7: **LOOP UNROLLING**

AIM:

To implement loop unrolling using LLVM.

CODING

```
#include<stdio.h>
int main()
{
int sum=0;
for(int i=0;i<100;i++)
{
sum=sum+i;
}
printf("%d",sum);
}
```

1)clang -emit-llvm -S lu.c -o lu.ll -Xclang -disable-O0-optnone
2)opt -mem2reg -simplycfg -loops -lcssa -loop-simplify -loop-rotate -loopunroll -o unrolling.ll -unroll-count=2 -unroll-allow-partial -S loopunroll.ll

lu.ll

```
; ModuleID = 'lu.c'
source_filename = "lu.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  store i32 0, i32* %2, align 4
  store i32 0, i32* %3, align 4
  br label %4
```

```
4:                                                ; preds = %11, %0
  %5 = load i32, i32* %3, align 4
  %6 = icmp slt i32 %5, 100
  br i1 %6, label %7, label %14
7:                                                ; preds = %4
  %8 = load i32, i32* %2, align 4
  %9 = load i32, i32* %3, align 4
  %10 = add nsw i32 %8, %9
  store i32 %10, i32* %2, align 4
  br label %11
11:                                               ; preds = %7
  %12 = load i32, i32* %3, align 4
  %13 = add nsw i32 %12, 1
  store i32 %13, i32* %3, align 4
  br label %4, !llvm.loop !6
14:                                               ; preds = %4
  %15 = load i32, i32* %2, align 4
  %16 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([3 x i8], [3
x i8]* @.str, i64 0, i64 0), i32 noundef %15)
  %17 = load i32, i32* %1, align 4
  ret i32 %17
}
declare i32 @printf(i8* noundef, ...) #1
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic" }
attributes #1 = { "frame-pointer"="all" "no-trapping-math"="true" "stack-
protector-buffer-size"="8" "target-cpu"="x86-64" "target-
features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
!6 = distinct !{!6, !7}
!7 = !{!"llvm.loop.mustprogress"}
```

lu-test.ll

```
; ModuleID = 'lu.c'
source_filename = "lu.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1

; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  store i32 0, i32* %2, align 4
  store i32 0, i32* %3, align 4
  br label %4
4:                                ; preds = %11, %0
  %5 = load i32, i32* %3, align 4
  %6 = icmp slt i32 %5, 100
  br i1 %6, label %7, label %14
7:                                ; preds = %4
  %8 = load i32, i32* %2, align 4
  %9 = load i32, i32* %3, align 4
  %10 = add nsw i32 %8, %9
  store i32 %10, i32* %2, align 4
  br label %11
11:                               ; preds = %7
  %12 = load i32, i32* %3, align 4
  %13 = add nsw i32 %12, 1
  store i32 %13, i32* %3, align 4
  br label %4, !llvm.loop !6
14:                               ; preds = %4
  %15 = load i32, i32* %2, align 4
  %16 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([3 x i8], [3
x i8]* @.str, i64 0, i64 0), i32 noundef %15)
  %17 = load i32, i32* %1, align 4
  ret i32 %17
}
```

```
declare i32 @printf(i8* noundef, ...) #1
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
attributes #1 = { "frame-pointer"="all" "no-trapping-math"="true" "stack-protector-buffer-size"="8" "target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
```

RESULT :

Thus loop unrolling has been implemented successfully.

Exercise 8: TAIL CALL ELIMINATION

AIM:

To implement tail call elimination using LLVM.

CODING

```
#include<stdio.h>
int tail(int n)
{
if(n>10)
return 10;
else
return 5;
}
int main()
{
int a=10;
return tail(a);
}
```

1)clang -emit-llvm -S tce.c -o tce.ll -Xclang -disable-O0-optnone
2)opt -mem2reg tce.ll -S -o tce-test.ll

tce.ll

```
; ModuleID = 'tce.c'
source_filename = "tce.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @tail(i32 noundef %0) #0 {
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 %0, i32* %3, align 4
  %4 = load i32, i32* %3, align 4
  %5 = icmp sgt i32 %4, 10
  br i1 %5, label %6, label %7
6:                                ; preds = %1
  store i32 10, i32* %2, align 4
```

```
  br label %8
7:                                ; preds = %1
  store i32 5, i32* %2, align 4
  br label %8
8:                                ; preds = %7, %6
  %9 = load i32, i32* %2, align 4
  ret i32 %9
}
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  store i32 0, i32* %1, align 4
  store i32 10, i32* %2, align 4
  %3 = load i32, i32* %2, align 4
  %4 = call i32 @tail(i32 noundef %3)
  ret i32 %4
}
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic"
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
```

tce-test.ll

```
; ModuleID = 'tce.ll'
source_filename = "tce.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @tail(i32 noundef %0) #0 {
  %2 = icmp sgt i32 %0, 10
```

```
  br i1 %2, label %3, label %4
3:                                ; preds = %1
  br label %5
4:                                ; preds = %1
  br label %5
5:                                ; preds = %4, %3
  %.0 = phi i32 [ 10, %3 ], [ 5, %4 ]
  ret i32 %.0
}
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = call i32 @tail(i32 noundef 10)
  ret i32 %1
}
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic" }
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
```

RESULT :

Thus tail call elimination has been implemented successfully.

Exercise 9: ALIAS ANALYSIS AND CAPTURING

AIM:

To implement alias analysis and capturing using LLVM.

CODING

```
#include<stdio.h>
int main()
{
int n=10;
int *p;
int *q;
int *s;
p=&n;
q=&n;
s=&n;
printf("%d",*p);
printf("%d",*q);
printf("%d",*s);
return 0;
}
```

1)clang -emit-llvm -S alias.c -o alias.ll -Xclang -disable-O0-optnone
2)opt -mem2reg --print-alias-sets alias.ll -S -o alias-test.ll

alias.ll

```
; ModuleID = 'alias.c'
source_filename = "alias.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32*, align 8
  %4 = alloca i32*, align 8
  %5 = alloca i32*, align 8
  store i32 0, i32* %1, align 4
```

```
  store i32 10, i32* %2, align 4
  store i32* %2, i32** %3, align 8
  store i32* %2, i32** %4, align 8
  store i32* %2, i32** %5, align 8
  %6 = load i32*, i32** %3, align 8
  %7 = load i32, i32* %6, align 4
  %8 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([3 x i8], [3
x i8]* @.str, i64 0, i64 0), i32 noundef %7)
  %9 = load i32*, i32** %4, align 8
  %10 = load i32, i32* %9, align 4
  %11 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([3 x i8], [3
x i8]* @.str, i64 0, i64 0), i32 noundef %10)
  %12 = load i32*, i32** %5, align 8
  %13 = load i32, i32* %12, align 4
  %14 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([3 x i8], [3
x i8]* @.str, i64 0, i64 0), i32 noundef %13)
  ret i32 0
}
declare i32 @printf(i8* noundef, ...) #1
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic" }
attributes #1 = { "frame-pointer"="all" "no-trapping-math"="true" "stack-
protector-buffer-size"="8" "target-cpu"="x86-64" "target-
features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
```

alias-test.ll

```
; ModuleID = 'alias.ll'
source_filename = "alias.c"
target datalayout = "e-m:e-p270:32:32-p271:32:32-p272:64:64-i64:64-f80:128-
n8:16:32:64-S128"
target triple = "x86_64-pc-linux-gnu"
@.str = private unnamed_addr constant [3 x i8] c"%d\00", align 1
; Function Attrs: noinline nounwind uwtable
define dso_local i32 @main() #0 {
  %1 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([3 x i8], [3
x i8]* @.str, i64 0, i64 0), i32 noundef 10)
  %2 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([3 x i8], [3
x i8]* @.str, i64 0, i64 0), i32 noundef 10)
  %3 = call i32 (i8*, ...) @printf(i8* noundef getelementptr inbounds ([3 x i8], [3
x i8]* @.str, i64 0, i64 0), i32 noundef 10)
  ret i32 0
}
declare i32 @printf(i8* noundef, ...) #1
attributes #0 = { noinline nounwind uwtable "frame-pointer"="all" "min-legal-
vector-width"="0" "no-trapping-math"="true" "stack-protector-buffer-size"="8"
"target-cpu"="x86-64" "target-features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87"
"tune-cpu"="generic" }
attributes #1 = { "frame-pointer"="all" "no-trapping-math"="true" "stack-
protector-buffer-size"="8" "target-cpu"="x86-64" "target-
features"="+cx8,+fxsr,+mmx,+sse,+sse2,+x87" "tune-cpu"="generic" }
!llvm.module.flags = !{!0, !1, !2, !3, !4}
!llvm.ident = !{!5}
!0 = !{i32 1, !"wchar_size", i32 4}
!1 = !{i32 7, !"PIC Level", i32 2}
!2 = !{i32 7, !"PIE Level", i32 2}
!3 = !{i32 7, !"uwtable", i32 1}
!4 = !{i32 7, !"frame-pointer", i32 2}
!5 = !{!"Ubuntu clang version 14.0.0-1ubuntu1"}
```

RESULT :

Thus alias analysis and capturing has been implemented successfully.