Dead Loop Deletion Pass

projekat na predmetu Konstrukcija kompilatora

Maša Cucić, 34/2020
Marija Grujičić, 35/2017

Svrha

- brisanje beskorisnih petlji ili petlji koje se ne izvršavaju
- analiza i optimizacija koda
- generisanje efikasnijeg izvršnog koda

Cilj

- smanjivanje broja instrukcija
- pojednostavljivanje IR koda

Koje petlje nisu korisne

- petlje koje se nikad ne izvršavaju (npr. brojač je 0)
- petlje koje nemaju neku korisnu instrukciju (npr. petlje koje imajuinstrukcije sa aritmetičkim operacijama ili poziv neke funkcije su korisne)
- petlje koje imaju beskorisne instrukcije (npr. množenje sa 1, deljenje sa 0, sabiranje sa 0
- petlje u kojima su svi operandi invarijantni (ne menjaju se) u svakoj iteraciji

Pokretanje pass-a

pass se pokreće iz terminala pozivanjem sledeće komande nakon pozicioniranja u build direktorijumu:

```
./bin/opt -load lib/LLVMDeadLoopDeletionPass.so -enable-new-pm=0
                    -dead-loops 1.ll -S -o output.ll
```

pri čemu je 1.ll IR kod generisan komandom:

```
./bin/clang -S -emit-llvm 1.c
```

PRIMERI

ako imamo naredni program:

```
int main(){
    int x = 0;

    // brise - OK
    for(int i=0; i< 10; i++){
     x *= 1;
    }
    return 0;
}
```

IR kod pre izvršavanja optimizacije možemo videti na slici pored

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  store i32 0, ptr %3, align 4
  br label %4

4:                                                ; preds = %10, %0
  %5 = load i32, ptr %3, align 4
  %6 = icmp slt i32 %5, 10
  br i1 %6, label %7, label %13

7:                                                ; preds = %4
  %8 = load i32, ptr %2, align 4
  %9 = mul nsw i32 %8, 1
  store i32 %9, ptr %2, align 4
  br label %10

10:                                               ; preds = %7
  %11 = load i32, ptr %3, align 4
  %12 = add nsw i32 %11, 1
  store i32 %12, ptr %3, align 4
  br label %4, !llvm.loop !6

13:                                               ; preds = %4
  ret i32 0
}
```

i nakon izvršavanja optimizacije:

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  store i32 0, ptr %3, align 4
  br label %4

4:                                                ; preds = %0
  ret i32 0
}
```

PRIMERI

ako imamo naredni program:

```
#include <stdio.h>

int main(){
    int x = 0;

    // brise - OK
    do {

    } while(x < 30);

}
```

IR kod pre izvršavanja optimizacije možemo videti na slici pored

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  br label %3

3:                                                ; preds = %4, %0
  br label %4

4:                                                ; preds = %3
  %5 = load i32, ptr %2, align 4
  %6 = icmp slt i32 %5, 30
  br i1 %6, label %3, label %7, !llvm.loop !6

7:                                                ; preds = %4
  %8 = load i32, ptr %1, align 4
  ret i32 %8
}
```

i nakon izvršavanja optimizacije:

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  br label %3

3:                                                ; preds = %0
  %4 = load i32, ptr %1, align 4
  ret i32 %4
}
```

PRIMERI

ako imamo naredni program:

```
#include <stdio.h>
int main(){
    int x = 0;
    // brise je - OK
    while(x < 0)
    {
        printf("brise");
    }

    return 0;
}
```

IR kod pre izvršavanja optimizacije možemo videti na slici pored

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  br label %3

3:                                                ; preds = %6, %0
  %4 = load i32, ptr %2, align 4
  %5 = icmp slt i32 %4, 0
  br i1 %5, label %6, label %8

6:                                                ; preds = %3
  %7 = call i32 (ptr, ...) @printf(ptr noundef @.str)
  br label %3, !llvm.loop !6

8:                                                ; preds = %3
  ret i32 0
}
```

i nakon izvršavanja optimizacije:

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  br label %3

3:                                                ; preds = %0
  ret i32 0
}
```

PRIMERI

ako imamo naredni program:

```
#include <stdio.h>

int main()
{
    // brise sve
    for(int j = 0; j < 0; j++)
    {
        printf("");

        for(int i = j + 1; i < 10; i++)
        {
            printf("ll");
        }
    }

    return 0;
}
```

IR kod pre izvršavanja optimizacije možemo videti na slici pored

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  br label %4

4:                                                ; preds = %20, %0
  %5 = load i32, ptr %2, align 4
  %6 = icmp slt i32 %5, 0
  br i1 %6, label %7, label %23

7:                                                ; preds = %4
  %8 = call i32 (ptr, ...) @printf(ptr noundef @.str)
  %9 = load i32, ptr %2, align 4
  %10 = add nsw i32 %9, 1
  store i32 %10, ptr %3, align 4
  br label %11

11:                                               ; preds = %16, %7
  %12 = load i32, ptr %3, align 4
  %13 = icmp slt i32 %12, 10
  br i1 %13, label %14, label %19

14:                                               ; preds = %11
  %15 = call i32 (ptr, ...) @printf(ptr noundef @.str.1)
  br label %16

16:                                               ; preds = %14
  %17 = load i32, ptr %3, align 4
  %18 = add nsw i32 %17, 1
  store i32 %18, ptr %3, align 4
  br label %11, !llvm.loop !6

19:                                               ; preds = %11
  br label %20

20:                                               ; preds = %19
  %21 = load i32, ptr %2, align 4
  %22 = add nsw i32 %21, 1
  store i32 %22, ptr %2, align 4
  br label %4, !llvm.loop !8

23:                                               ; preds = %4
  ret i32 0
}
```

i nakon izvršavanja optimizacije:

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  br label %4

4:                                                ; preds = %0
  ret i32 0
}
```

PRIMERI

ako imamo naredni program:

```
#include <stdio.h>

int main() {

    int x = 0;
    int y = 3;

    // ne brise - OK
    do {
      if(0)
        x++;
      y += 3;
    } while(x < 10);

}
```

IR kod pre izvršavanja optimizacije možemo videti na slici pored

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  store i32 3, ptr %3, align 4
  br label %4

4:                                                ; preds = %7, %0
  %5 = load i32, ptr %3, align 4
  %6 = add nsw i32 %5, 3
  store i32 %6, ptr %3, align 4
  br label %7

7:                                                ; preds = %4
  %8 = load i32, ptr %2, align 4
  %9 = icmp slt i32 %8, 10
  br i1 %9, label %4, label %10, !llvm.loop !6

10:                                               ; preds = %7
  %11 = load i32, ptr %1, align 4
  ret i32 %11
}
```

i nakon izvršavanja optimizacije:

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  store i32 3, ptr %3, align 4
  br label %4

4:                                                ; preds = %7,
%0
  %5 = load i32, ptr %3, align 4
  %6 = add nsw i32 %5, 3
  store i32 %6, ptr %3, align 4
  br label %7

7:                                                ; preds = %4
  %8 = load i32, ptr %2, align 4
  %9 = icmp slt i32 %8, 10
  br i1 %9, label %4, label %10, !llvm.loop !6

10:                                               ; preds = %7
  %11 = load i32, ptr %1, align 4
  ret i32 %11
}
```

PRIMERI

ako imamo naredni program:

```
#include <stdio.h>

int main() {

    int x = 0;
    int y = 3;

    // brise - OK
    do {
        if(0)
            x++;
    } while(x < 10);
}
```

IR kod pre izvršavanja optimizacije možemo videti na slici pored

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  store i32 3, ptr %3, align 4
  br label %4

4:                                                ; preds = %5, %0
  br label %5

5:                                                ; preds = %4
  %6 = load i32, ptr %2, align 4
  %7 = icmp slt i32 %6, 10
  br i1 %7, label %4, label %8, !llvm.loop !6

8:                                                ; preds = %5
  %9 = load i32, ptr %1, align 4
  ret i32 %9
}
```

i nakon izvršavanja optimizacije:

```
define dso_local i32 @main() #0 {
  %1 = alloca i32, align 4
  %2 = alloca i32, align 4
  %3 = alloca i32, align 4
  store i32 0, ptr %1, align 4
  store i32 0, ptr %2, align 4
  store i32 3, ptr %3, align 4
  br label %4

4:                                                ; preds = %0
  %5 = load i32, ptr %1, align 4
  ret i32 %5
}
```

Kraj