Konstrukcija kompilatora

LLVM IR

Faktorijel

```
1  int factorial(int val);
2
3  int main()
4  {
5      return factorial(2) * 7 == 42;
6  }
```

Faktorijel IR

```
1 declare i32 @factorial(i32)
```

Faktorijel IR

```
1 declare i32 @factorial(i32)
2 define i32 @main(i32 %argc, i8** %argv) {
```

Faktorijel IR

```
1  declare i32 @factorial(i32)
2  define i32 @main(i32 %argc, i8** %argv) {
3      %1 = call i32 @factorial(i32 2)
```

Faktorijel IR

```
1  declare i32 @factorial(i32)
2  define i32 @main(i32 %argc, i8** %argv) {
3      %1 = call i32 @factorial(i32 2)
4      %2 = mul i32 %1, 7
```

Faktorijel IR

```
1  declare i32 @factorial(i32)
2  define i32 @main(i32 %argc, i8** %argv) {
3      %1 = call i32 @factorial(i32 2)
4      %2 = mul i32 %1, 7
5      %3 = icmp eq i32 %2, 42
```

Faktorijel IR

```
1  declare i32 @factorial(i32)
2  define i32 @main(i32 %argc, i8** %argv) {
3      %1 = call i32 @factorial(i32 2)
4      %2 = mul i32 %1, 7
5      %3 = icmp eq i32 %2, 42
6      %result = zext i1 %3 to i32
```

Faktorijel IR

```
1  declare i32 @factorial(i32)
2  define i32 @main(i32 %argc, i8** %argv) {
3      %1 = call i32 @factorial(i32 2)
4      %2 = mul i32 %1, 7
5      %3 = icmp eq i32 %2, 42
6      %result = zext i1 %3 to i32
7      ret i32 %result
8  }
```

Faktorijel implemetacija

```
1  int factorial(int val)
2  {
3      if (val == 0) {
4          return 1;
5      }
6      return val * factorial(val - 1);
7  }
```

Faktorijel implemetacija IR

```
1  define i32 @factorial(i32 %val) {
```

Faktorijel implemetacija IR

```
1 define i32 @factorial(i32 %val) {
2     %is_base_case = icmp eq i32 %val, 0
3     br i1 %is_base_case, label %base_case, label %rec_case
```

Faktorijel implemetacija IR

```
1 define i32 @factorial(i32 %val) {
2     %is_base_case = icmp eq i32 %val, 0
3     br i1 %is_base_case, label %base_case, label %rec_case
4
5     base_case:
6         ret i32 1
```

Faktorijel implemetacija IR

```
1 define i32 @factorial(i32 %val) {
2     %is_base_case = icmp eq i32 %val, 0
3     br i1 %is_base_case, label %base_case, label %rec_case
4
5     base_case:
6         ret i32 1
7
8     rec_case:
9         %1 = add i32 -1, %val
```

Faktorijel implemetacija IR

```
1  define i32 @factorial(i32 %val) {
2      %is_base_case = icmp eq i32 %val, 0
3      br i1 %is_base_case, label %base_case, label %rec_case
4
5  base_case:
6      ret i32 1
7
8  rec_case:
9      %1 = add i32 -1, %val
0      %2 = call i32 @factorial(i32 %1)
```

Faktorijel implemetacija IR

```
1  define i32 @factorial(i32 %val) {
2      %is_base_case = icmp eq i32 %val, 0
3      br i1 %is_base_case, label %base_case, label %rec_case
4
5  base_case:
6      ret i32 1
7
8  rec_case:
9      %1 = add i32 -1, %val
0      %2 = call i32 @factorial(i32 %1)
1      %3 = mul i32 %val, %2
```

Faktorijel implemetacija IR

```
1  define i32 @factorial(i32 %val) {
2      %is_base_case = icmp eq i32 %val, 0
3      br i1 %is_base_case, label %base_case, label %rec_case
4
5  base_case:
6      ret i32 1
7
8  rec_case:
9      %1 = add i32 -1, %val
0      %2 = call i32 @factorial(i32 %1)
1      %3 = mul i32 %val, %2
2      ret i32 %3
3  }
```

Faktorijel implemetacija IR

```
1  define i32 @factorial(i32 %val) {
2  entry:
3      %is_base_case = icmp eq i32 %val, 0
4      br i1 %is_base_case, label %base_case, label %rec_case
5
6  base_case:
7      ret i32 1
8
9  rec_case:
0      %0 = add i32 -1, %val
1      %1 = call i32 @factorial(i32 %1)
2      %2 = mul i32 %val, %2
3      ret i32 %2
4  }
```

Faktorijel iterativno

```
1  int facotrial(int val)
2  {
3      int temp = 1;
4      for (int i = 2; i <= val; i++) {
5          temp *= i;
6      }
7      return temp;
8  }
```

Faktorijel iterativno IR

```
1                define i32 @factorial(i32 %val) {
2                entry:
3                    %i = add i32 0, 2
4                    %temp = add i32 0, 1
5                    br label %check_for_condition
```

Faktorijel iterativno IR

```
1     define i32 @factorial(i32 %val) {
2     entry:
3         %i = add i32 0, 2
4         %temp = add i32 0, 1
5         br label %check_for_condition
6
7     check_for_condition:
8         %i_leq_val = icmp sle i32 %i, %val
9         br i1 %i_leq_val, label %for_body, label %end_loop
```

Faktorijel iterativno IR

```
1   define i32 @factorial(i32 %val) {
2   entry:
3       %i = add i32 0, 2
4       %temp = add i32 0, 1
5       br label %check_for_condition
6
7   check_for_condition:
8       %i_leq_val = icmp sle i32 %i, %val
9       br i1 %i_leq_val, label %for_body, label %end_loop
0
1   end_loop:
2       ret i32 %temp
3   }
```

Faktorijel iterativno IR

```
1  define i32 @factorial(i32 %val) {
2  entry:
3      %i = add i32 0, 2
4      %temp = add i32 0, 1
5      br label %check_for_condition
6
7  check_for_condition:
8      %i_leq_val = icmp sle i32 %i, %val
9      br i1 %i_leq_val, label %for_body, label %end_loop
10
11 for_body:
12     %temp = mul i32 %temp, i
13     %i = add i32 %i, 1
14     br check_for_condition
15
16 end_loop:
17     ret i32 %temp
18 }
```

SSA - Static Single Assignment

Faktorijel iterativno IR

```
1  define i32 @factorial(i32 %val) {
2  entry:
3      %i = add i32 0, 2
4      %temp = add i32 0, 1
5      br label %check_for_condition
6
7  check_for_condition:
8      %i_leq_val = icmp sle i32 %i, %val
9      br i1 %i_leq_val, label %for_body, label %end_loop
10
11 for_body:
12     %new_temp = mul i32 %temp, i
13     %i_plus_one = add i32 %i, 1
14     br check_for_condition
15
16 end_loop:
17     ret i32 %temp
18 }
```

Phi instrukcije

<result> = phi <ty> [<val0>, <label0>], [<val1, label1>]

Faktorijel iterativno IR

```
1  define i32 @factorial(i32 %val) {
2  entry:
3      br label %check_for_condition
4
5  check_for_condition:
6      %curr_i = phi i32 [2, %entry], [%i_plus_one, %for_body]
7      %temp = phi i32 [1, %entry], [%new_temp, %for_body]
8      %i_leq_val = icmp sle i32 %curr_i, %val
9      br i1 %i_leq_val, label %for_body, label %end_loop
0
1  for_body:
2      %new_temp = mul i32 %temp, curr_i
3      %i_plus_one = add i32 %curr_i, 1
4      br check_for_condition
5
6  end_loop:
7      ret i32 %temp
8  }
```

Faktorijel iterativno IR

```
1    define i32 @factorial(i32 %val) {
2    entry:
3        %i.addr = alloca i32
4        %temp.addr = alloca i32
```

Faktorijel iterativno IR

```
1     define i32 @factorial(i32 %val) {
2     entry:
3         %i.addr = alloca i32
4         %temp.addr = alloca i32
5         store i32 2, i32* %i.addr
6         store i32 1, i32* %temp.addr
```

Faktorijel iterativno IR

```
1     define i32 @factorial(i32 %val) {
2     entry:
3         %i.addr = alloca i32
4         %temp.addr = alloca i32
5         store i32 2, i32* %i.addr
6         store i32 1, i32* %temp.addr
7         br label %check_for_condition
8
9     check_for_condition:
0         ----------------------------------------
1         %i_leq_val = icmp sle i32 %current_i, %val
2         br i1 %i_leq_val, label %for_body, label %end_loop
```

Faktorijel iterativno IR

```
1   define i32 @factorial(i32 %val) {
2   entry:
3       %i.addr = alloca i32
4       %temp.addr = alloca i32
5       store i32 2, i32* %i.addr
6       store i32 1, i32* %temp.addr
7       br label %check_for_condition
8
9   check_for_condition:
0       %current_i = load i32, i32* %i.addr
1       %temp = load i32, i32* %temp.addr
2       %i_leq_val = icmp sle i32 %current_i, %val
3       br i1 %i_leq_val, label %for_body, label %end_loop
```

Faktorijel iterativno IR

```
1   define i32 @factorial(i32 %val) {
2   entry:
3       %i.addr = alloca i32
4       %temp.addr = alloca i32
5       store i32 2, i32* %i.addr
6       store i32 1, i32* %temp.addr
7       br label %check_for_condition
8
9   check_for_condition:
0       %current_i = load i32, i32* %i.addr
1       %temp = load i32, i32* %temp.addr
2       %i_leq_val = icmp sle i32 %current_i, %val
3       br i1 %i_leq_val, label %for_body, label %end_loop
4
5   for_body:
6       %i_plus_one = add i32 %current_i, 1
7       %new_temp = mul i32 %temp, current_i
8       ----------------------------------------
9   end_loop:
0       ret i32 %temp
1   }
```

Faktorijel iterativno IR

```
1   define i32 @factorial(i32 %val) {
2   entry:
3       %i.addr = alloca i32
4       %temp.addr = alloca i32
5       store i32 2, i32* %i.addr
6       store i32 1, i32* %temp.addr
7       br label %check_for_condition
8
9   check_for_condition:
10      %current_i = load i32, i32* %i.addr
11      %temp = load i32, i32* %temp.addr
12      %i_leq_val = icmp sle i32 %current_i, %val
13      br i1 %i_leq_val, label %for_body, label %end_loop
14
15  for_body:
16      %i_plus_one = add i32 %current_i, 1
17      %new_temp = mul i32 %temp, current_i
18      store i32 %i_plus_one, i32* %i.addr
19      store i32 %new_temp, i32* %temp.addr
20      br check_for_condition
21
22  end_loop:
23      ret i32 %temp
24  }
```