


SWPP2023: optimization ideas

Team 2

```
1 BB_entry:
2     %x1 = add i32 1, 1
3     %x2 = mul i32 %x1, 7
4     %xcond = icmp sge i32 %x2, 10
5     br i1 %xcond, label %BB_always, label %BB_unreachable
6
7 BB_always:
8     %tmp1 = fadd double 3.14, 2.72
9     %val1 = mul i32 1, 1
10    br label %BB_end
11
12 BB_unreachable:
13    %tmp2 = fsub double 1.41, 1.62
14    %val2 = mul i32 2, 2
15    br label %BB_end
16
17 BB_end:
18    %tmp = phi double [%tmp1, %BB_always], [%tmp2, %BB_unreachable]
19    %val = phi i32 [%val1, %BB_always], [%val2, %BB_unreachable]
20    ret i32 %val
```

Detect Side Effects

- Remove instructions without side effects



```
1 BB_entry:
2     %x1 = add i32 1, 1
3     %x2 = mul i32 %x1, 7
4     %xcond = icmp sge i32 %x2, 10
5     br i1 %xcond, label %BB_always, label %BB_unreachable
6
7 BB_always:
8     %tmp1 = fadd double 3.14, 2.72
9     %val1 = mul i32 1, 1
10    br label %BB_end
11
12 BB_unreachable:
13    %tmp2 = fsub double 1.41, 1.62
14    %val2 = mul i32 2, 2
15    br label %BB_end
16
17 BB_end:
18    %tmp = phi double [%tmp1, %BB_always], [%tmp2, %BB_unreachable]
19    %val = phi i32 [%val1, %BB_always], [%val2, %BB_unreachable]
20    ret i32 %val
```

```
1  BB_entry:
2      %x1 = add i32 1, 1
3      %x2 = mul i32 %x1, 7
4      %xcond = icmp sge i32 %x2, 10
5      br i1 %xcond, label %BB_always, label %BB_unreachable
6
7  BB_always:
8      %tmp1 = fadd double 3.14, 2.72
9      %val1 = mul i32 1, 1
10     br label %BB_end
11
12  BB_unreachable:
13     %tmp2 = fsub double 1.41, 1.62
14     %val2 = mul i32 2, 2
15     br label %BB_end
16
17  BB_end:
18     %tmp = phi double [%tmp1, %BB_always], [%tmp2, %BB_unreachable]
19     %val = phi i32 [%val1, %BB_always], [%val2, %BB_unreachable]
20     ret i32 %val
```

Basic optimizations

- Detect constant expressions

```
1 BB_entry:
2     %x1 = add i32 1, 1
3     %x2 = mul i32 %x1, 7
4     %xcond = icmp sge i32 %x2, 10
5     br i1 %xcond, label %BB_always, label %BB_unreachable
6
7 BB_always:
8     %tmp1 = fadd double 3.14, 2.72
9     %val1 = mul i32 1, 1
10    br label %BB_end
11
12 BB_unreachable:
13    %tmp2 = fsub double 1.41, 1.62
14    %val2 = mul i32 2, 2
15    br label %BB_end
16
17 BB_end:
18    %tmp = phi double [%tmp1, %BB_always], [%tmp2, %BB_unreachable]
19    %val = phi i32 [%val1, %BB_always], [%val2, %BB_unreachable]
20    ret i32 %val
```

Basic optimizations

- Always-branch optimization


```
1 BB_entry:
2     %x1 = add i32 1, 1
3     %x2 = mul i32 %x1, 7
4     %xcond = icmp sge i32 %x2, 10
5     br i1 %xcond, label %BB_always, label %BB_unreachable
6
7 BB_always:
8     %tmp1 = fadd double 3.14, 2.72
9     %val1 = mul i32 1, 1
10    br label %BB_end
11
12 BB_unreachable:
13    %tmp2 = fsub double 1.41, 1.62
14    %val2 = mul i32 2, 2
15    br label %BB_end
16
17 BB_end:
18    %tmp = phi double [%tmp1, %BB_always], [%tmp2, %BB_unreachable]
19    %val = phi i32 [%val1, %BB_always], [%val2, %BB_unreachable]
20    ret i32 %val
```

Basic optimizations

- Delete unreachable

```
1 BB_entry:
2     %x1 = add i32 1, 1
3     %x2 = mul i32 %x1, 7
4     %xcond = icmp sge i32 %x2, 10
5     br i1 %xcond, label %BB_always, label %BB_unreachable
6
7 BB_always:
8     %tmp1 = fadd double 3.14, 2.72
9     %val1 = mul i32 1, 1
10    br label %BB_end
11
12 BB_unreachable:
13 %tmp2 = fsub double 1.41, 1.62
14 %val2 = mul i32 2, 2
15 br label %BB_end
16
17 BB_end:
18     %tmp = phi double [%tmp1, %BB_always], [%tmp2, %BB_unreachable]
19     %val = phi i32 [%val1, %BB_always], [%val2, %BB_unreachable]
20     ret i32 %val
```

Basic optimizations

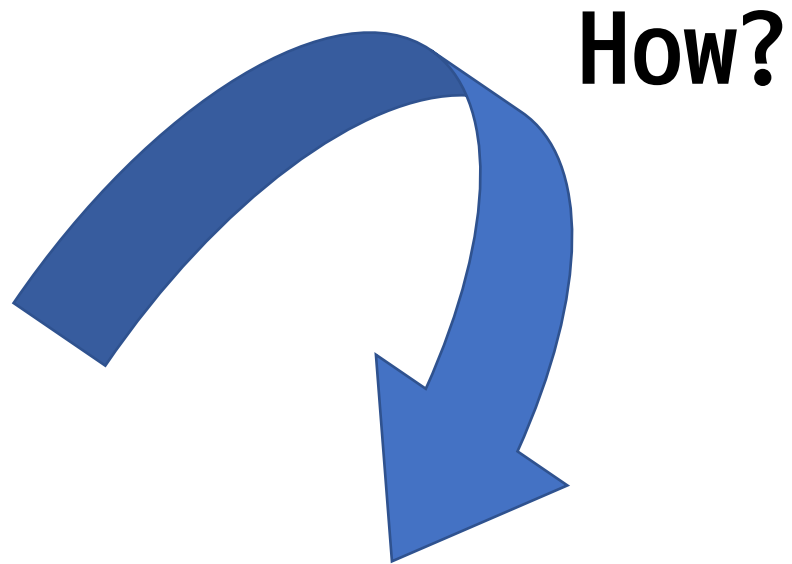
- Substitute instructions to cheaper equivalents

```
1 BB_entry:
2     %x1 = add i32 1, 1
3     %x2 = mul i32 %x1, 7
4     %xcond = icmp sge i32 %x2, 10
5     br i1 %xcond, label %BB_always, label %BB_unreachable
6
7 BB_always:
8     %tmp1 = fadd double 3.14, 2.72
9     %val1 = mul i32 1, 1
10    br label %BB_end
11
12 BB_unreachable:
13    %tmp2 = fsub double 1.41, 1.62
14    %val2 = mul i32 2, 2
15    br label %BB_end
16
17 BB_end:
18    %tmp = phi double [%tmp1, %BB_always], [%tmp2, %BB_unreachable]
19    %val = phi i32 [%val1, %BB_always], [%val2, %BB_unreachable]
20    ret i32 %val
```

Basic optimizations

- Repeat until no more optimization is possible

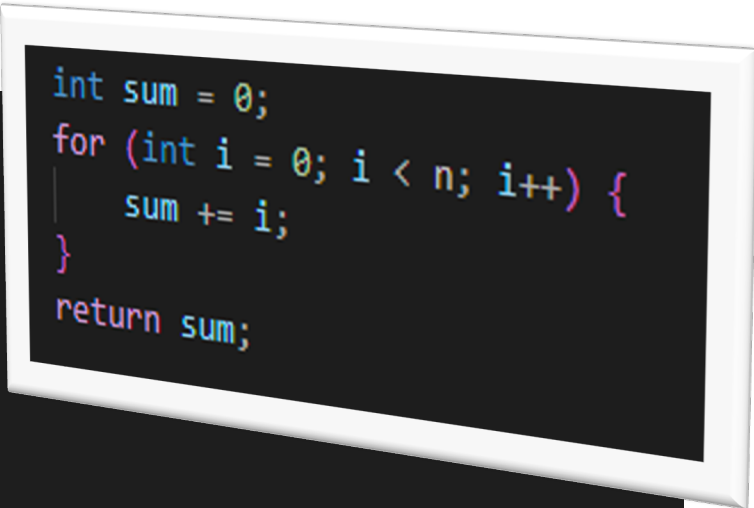
```
1 BB_entry:
2     br label %BB_always
3
4 BB_always:
5     %val1 = mul i32 1, 1
6     br label %BB_end
7
8 BB_end:
9     %val = mul i32 %val1, 1
10    ret i32 %val
```



```
1 BB_entry:
2     ret i32 1
```

Loop optimizations

- For-loop: set exit condition as true



```
int sum = 0;
for (int i = 0; i < n; i++) {
    sum += i;
}
return sum;
```

```
1  for.init:
2      %i.init = mul i32 1, 1
3      %sum.init = mul i32 0, 0
4      %cond.init = icmp ult i64 %i.init, %n
5      br i1 %cond.init, label for.body, label for.end
6
7  for.body:
8      %i = phi i32 [ %i.next, %for.body ], [ %i.init, %for.init ]
9      %sum = phi i32 [ %sum.next, %for.body ], [ %sum.init, %for.init ]
10     %sum.next = add i32 %sum, %i
11     %i.next = add i32 %i, 1
12     %cond = icmp ult i64 %i.next, %n
13     br i1 %cond, label for.body, label for.end
14
15  for.end:
16     %sum.ret = phi i32 [ %sum.next, %for.body ], [ %sum.init, %for.init ]
17     ret i32 %sum.ret
18
```

True for n times, false for only 1 time

```
int sum = 0;
for (int i = 0; !(i >= n); i++) {
    sum += i;
}
return sum;
```

```
1  for.init:
2      %i.init = mul i32 1, 1
3      %sum.init = mul i32 0, 0
4      %cond.init = icmp uge i64 %i.init, %n
5      br i1 %cond.init, label for.end, label for.body
6
7  for.body:
8      %i = phi i32 [ %i.next, %for.body ], [ %i.init, %for.init ]
9      %sum = phi i32 [ %sum.next, %for.body ], [ %sum.init, %for.init ]
10     %sum.next = add i32 %sum, %i
11     %i.next = add i32 %i, 1
12     %cond = icmp uge i64 %i.next, %n
13     br i1 %cond, label for.end, label for.body
14
15 for.end:
16     %sum.ret = phi i32 [ %sum.next, %for.body ], [ %sum.init, %for.init ]
17     ret i32 %sum.ret
18
```

Loop optimizations

- How to detect loop?

Other Possible Implementations?

- Usage of Oracle
- Put heap into stack