Team Idea Presentation

Team 4

김민석, 김준혁, 유준열, 최지훈

최적화 방법

- Memory access cost 최소화
- 연산 cost 최소화

Memory access cost 최소화

- 레지스터 개수 줄이기

```
%z = add %x, %y
use %x
use %z
```

```
use %x
%z = add %x, %y
use %z
```

(LLVM level)

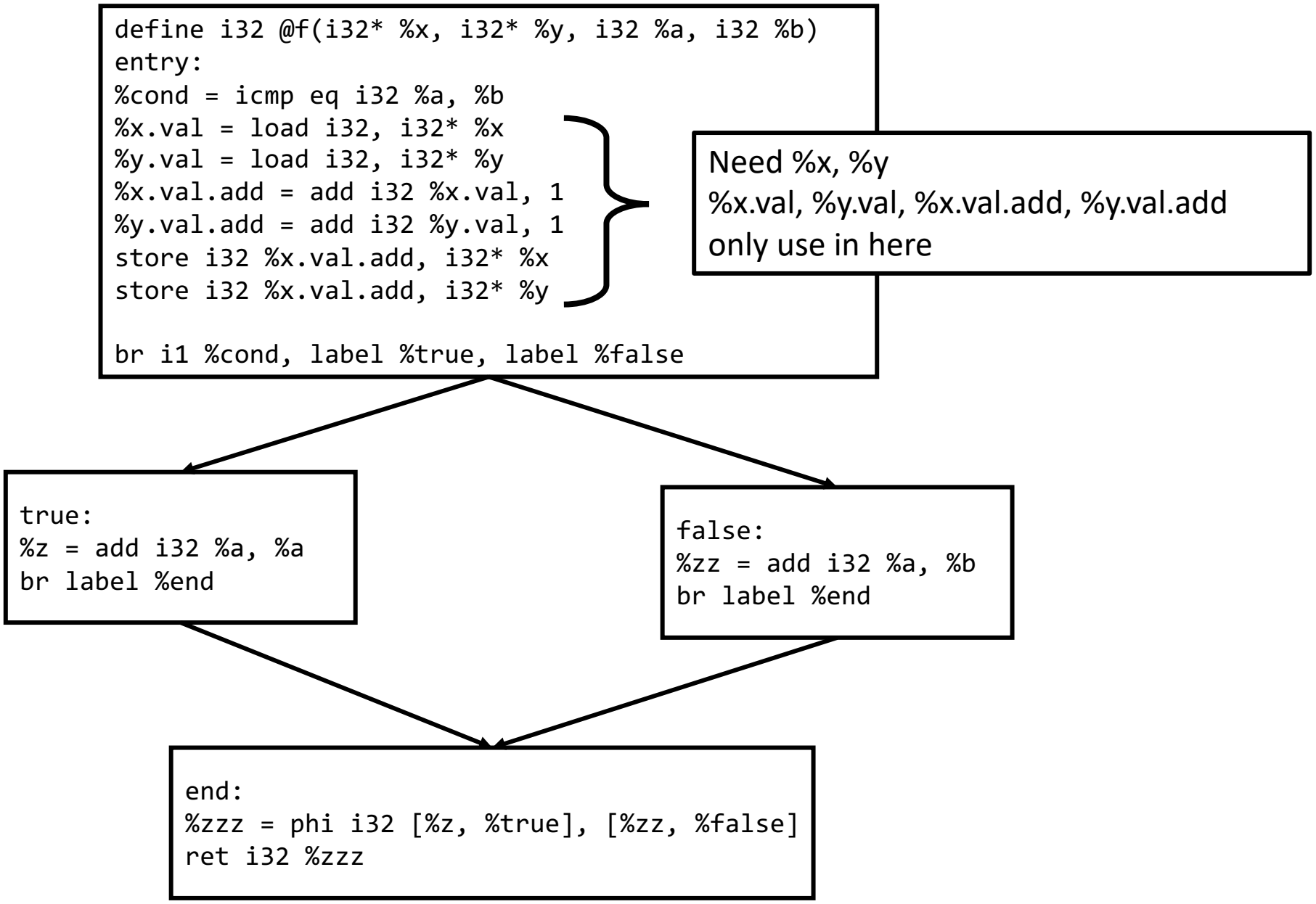
```
%r3 = add %r1, %r2
use %r1
use %r3
```

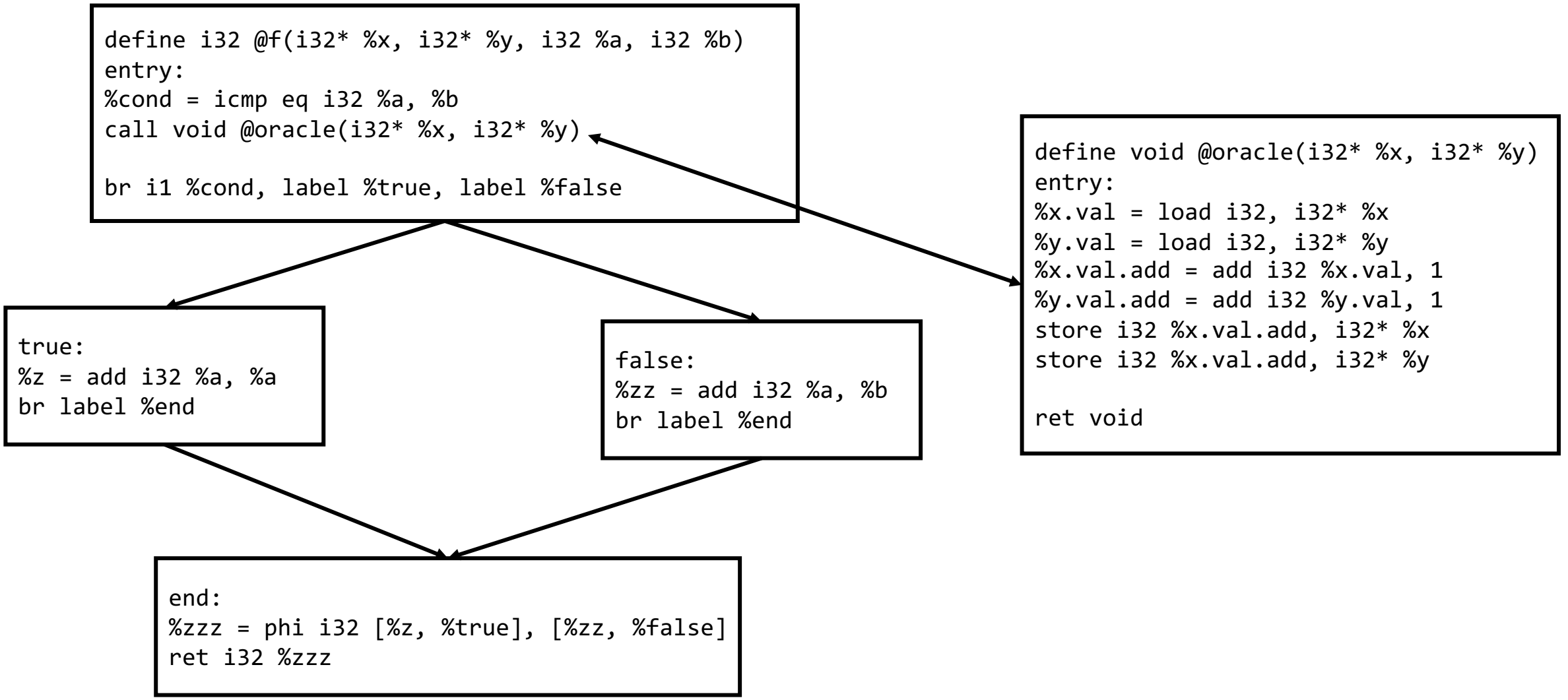
```
use %r1
%r1 = add %r1, %r2
use %r1
```

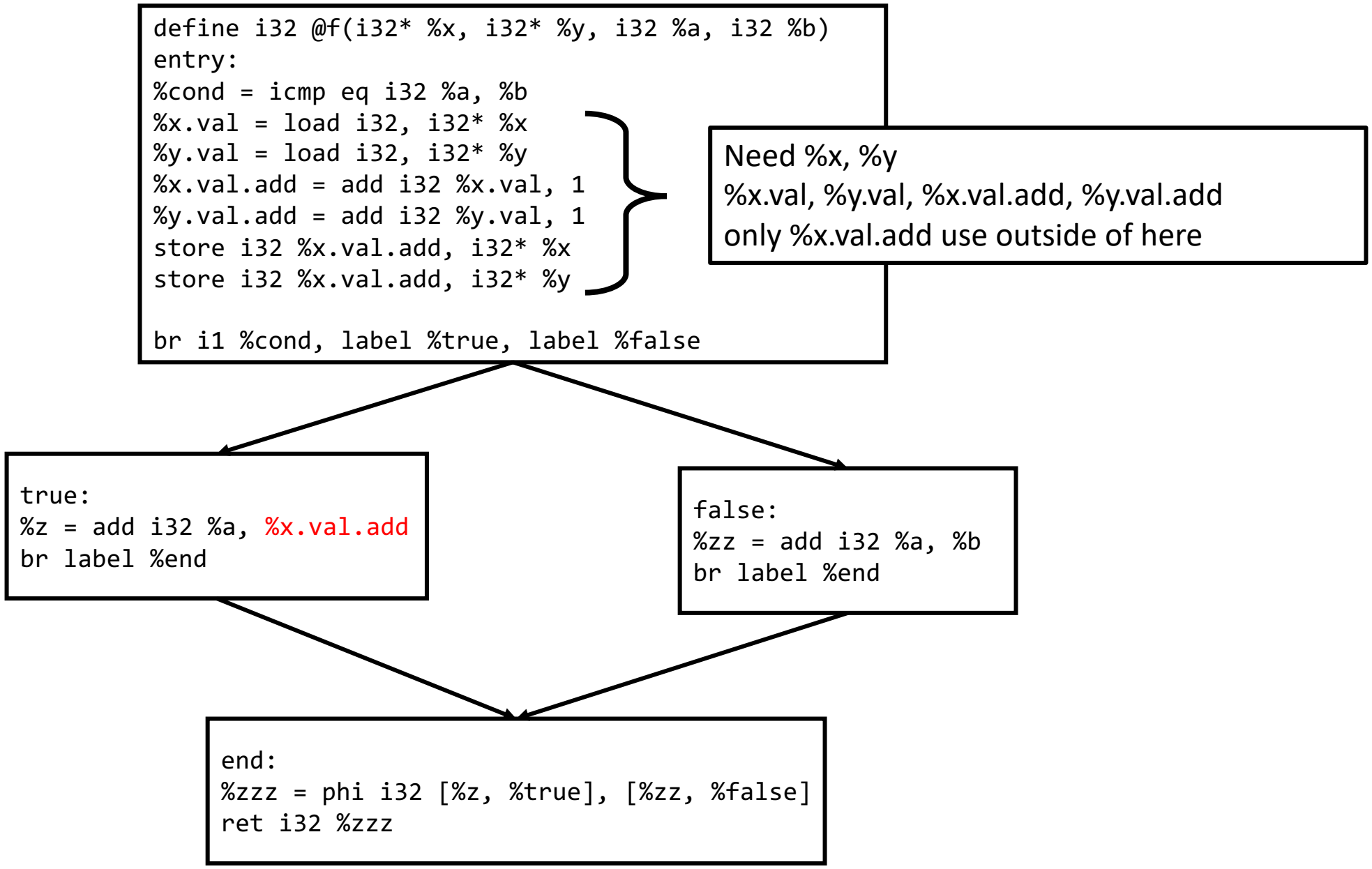
(어셈블리 level)

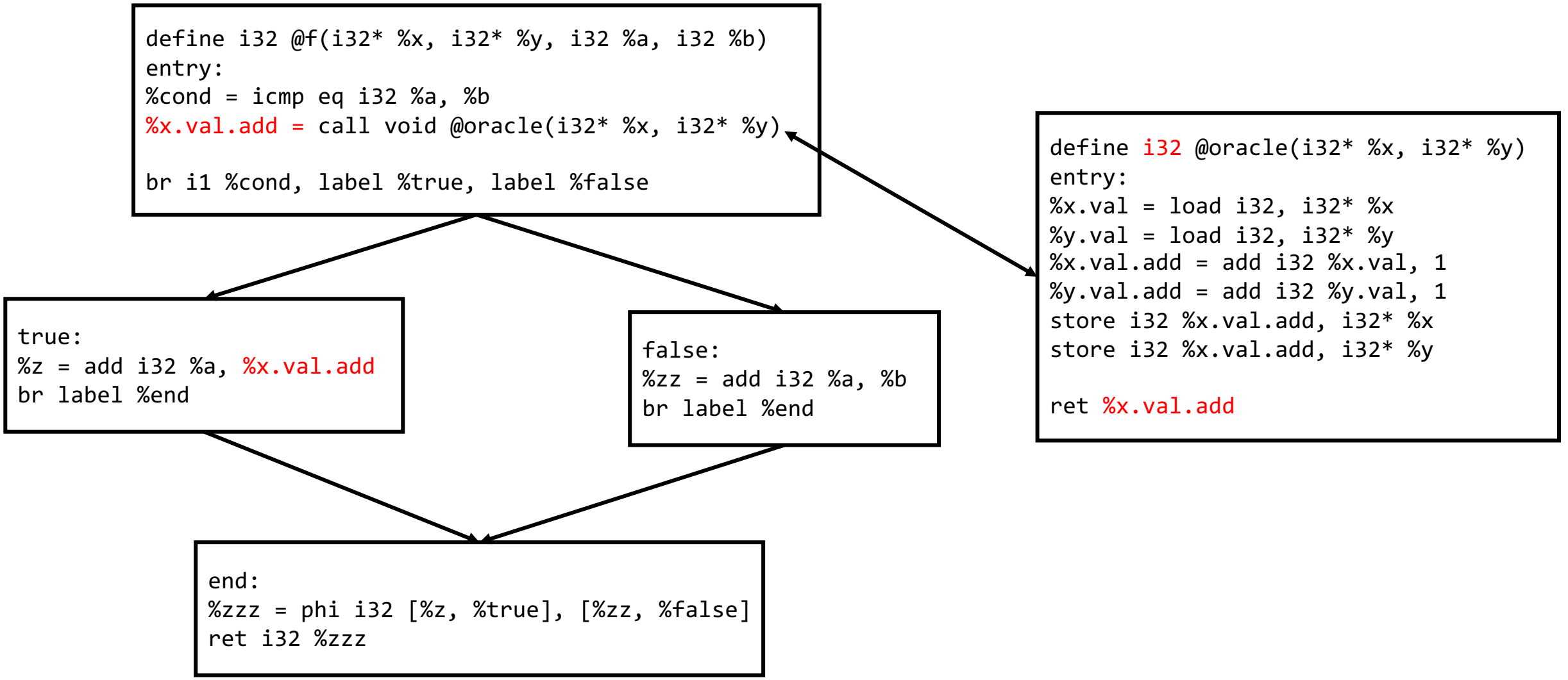
Memory access cost 최소화

- oracle을 이용한 load, store 뭉치기
  - 다수의 load, store가 있을 경우 oracle 함수를 이용하는 것이 효율적


define i32 @f(i32* %x, i32* %y, i32 %a, i32 %b)
entry:
%cond = icmp eq i32 %a, %b
%x.val = load i32, i32* %x
%y.val = load i32, i32* %y
%x.val.add = add i32 %x.val, 1
%y.val.add = add i32 %y.val, 1
store i32 %x.val.add, i32* %x
store i32 %x.val.add, i32* %y
br i1 %cond, label %true, label %false
Need %x, %y
%x.val, %y.val, %x.val.add, %y.val.add
only use in here
true:
%z = add i32 %a, %a
br label %end
false:
%zz = add i32 %a, %b
br label %end
end:
%zzz = phi i32 [%z, %true], [%zz, %false]
ret i32 %zzz


define i32 @f(i32* %x, i32* %y, i32 %a, i32 %b)
entry:
%cond = icmp eq i32 %a, %b
call void @oracle(i32* %x, i32* %y)
br i1 %cond, label %true, label %false
true:
%z = add i32 %a, %a
br label %end
false:
%zz = add i32 %a, %b
br label %end
end:
%zzz = phi i32 [%z, %true], [%zz, %false]
ret i32 %zzz
define void @oracle(i32* %x, i32* %y)
entry:
%x.val = load i32, i32* %x
%y.val = load i32, i32* %y
%x.val.add = add i32 %x.val, 1
%y.val.add = add i32 %y.val, 1
store i32 %x.val.add, i32* %x
store i32 %x.val.add, i32* %y
ret void

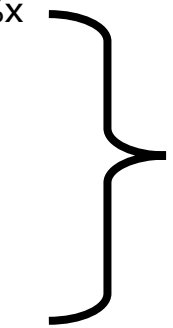

define i32 @f(i32* %x, i32* %y, i32 %a, i32 %b)
entry:
%cond = icmp eq i32 %a, %b
%x.val = load i32, i32* %x
%y.val = load i32, i32* %y
%x.val.add = add i32 %x.val, 1
%y.val.add = add i32 %y.val, 1
store i32 %x.val.add, i32* %x
store i32 %x.val.add, i32* %y
br i1 %cond, label %true, label %false
Need %x, %y
%x.val, %y.val, %x.val.add, %y.val.add
only %x.val.add use outside of here
true:
%z = add i32 %a, %x.val.add
br label %end
false:
%zz = add i32 %a, %b
br label %end
end:
%zzz = phi i32 [%z, %true], [%zz, %false]
ret i32 %zzz


define i32 @f(i32* %x, i32* %y, i32 %a, i32 %b)
entry:
%cond = icmp eq i32 %a, %b
%x.val.add = call void @oracle(i32* %x, i32* %y)
br i1 %cond, label %true, label %false
true:
%z = add i32 %a, %x.val.add
br label %end
false:
%zz = add i32 %a, %b
br label %end
end:
%zzz = phi i32 [%z, %true], [%zz, %false]
ret i32 %zzz
define i32 @oracle(i32* %x, i32* %y)
entry:
%x.val = load i32, i32* %x
%y.val = load i32, i32* %y
%x.val.add = add i32 %x.val, 1
%y.val.add = add i32 %y.val, 1
store i32 %x.val.add, i32* %x
store i32 %x.val.add, i32* %y
ret %x.val.add

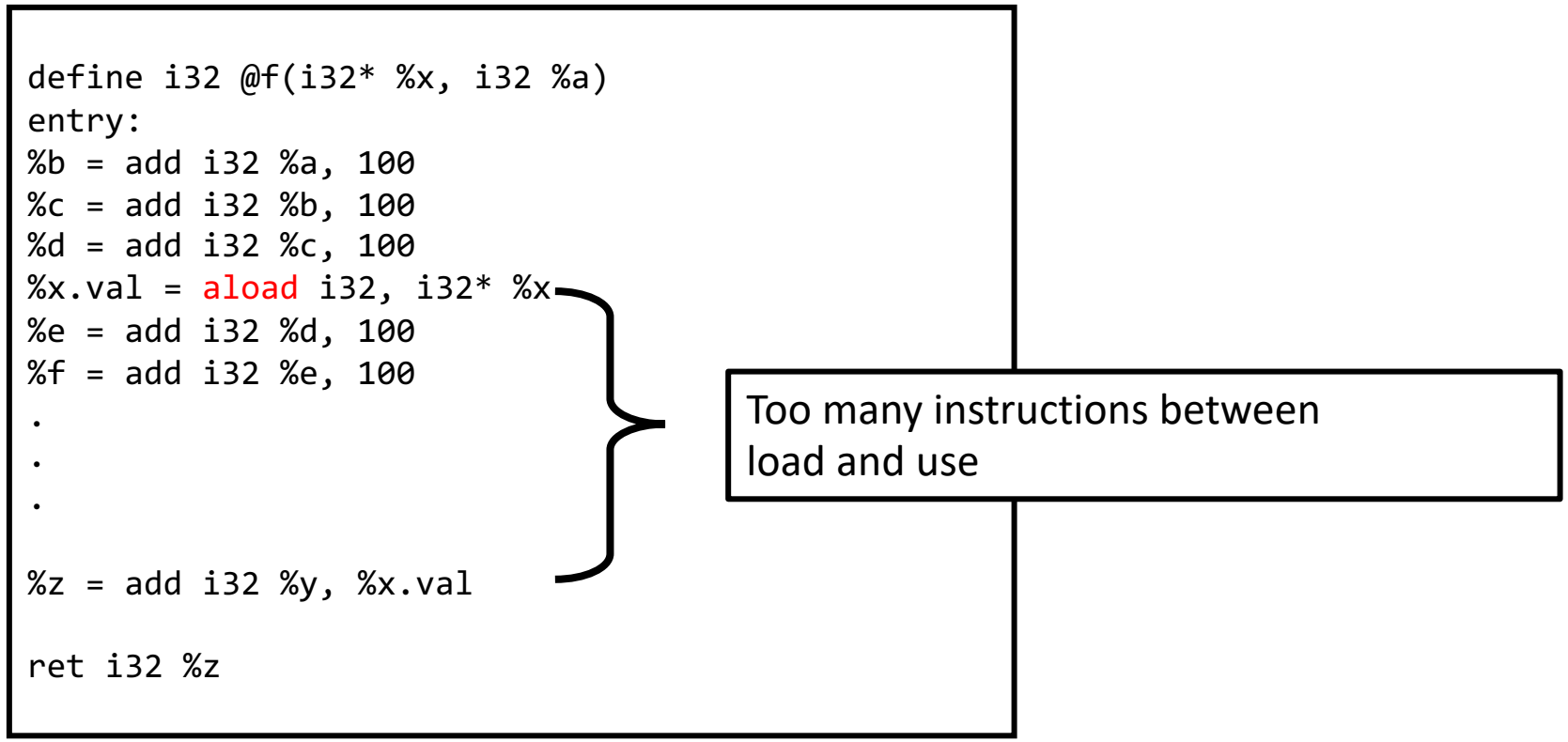
Memory access cost 최소화

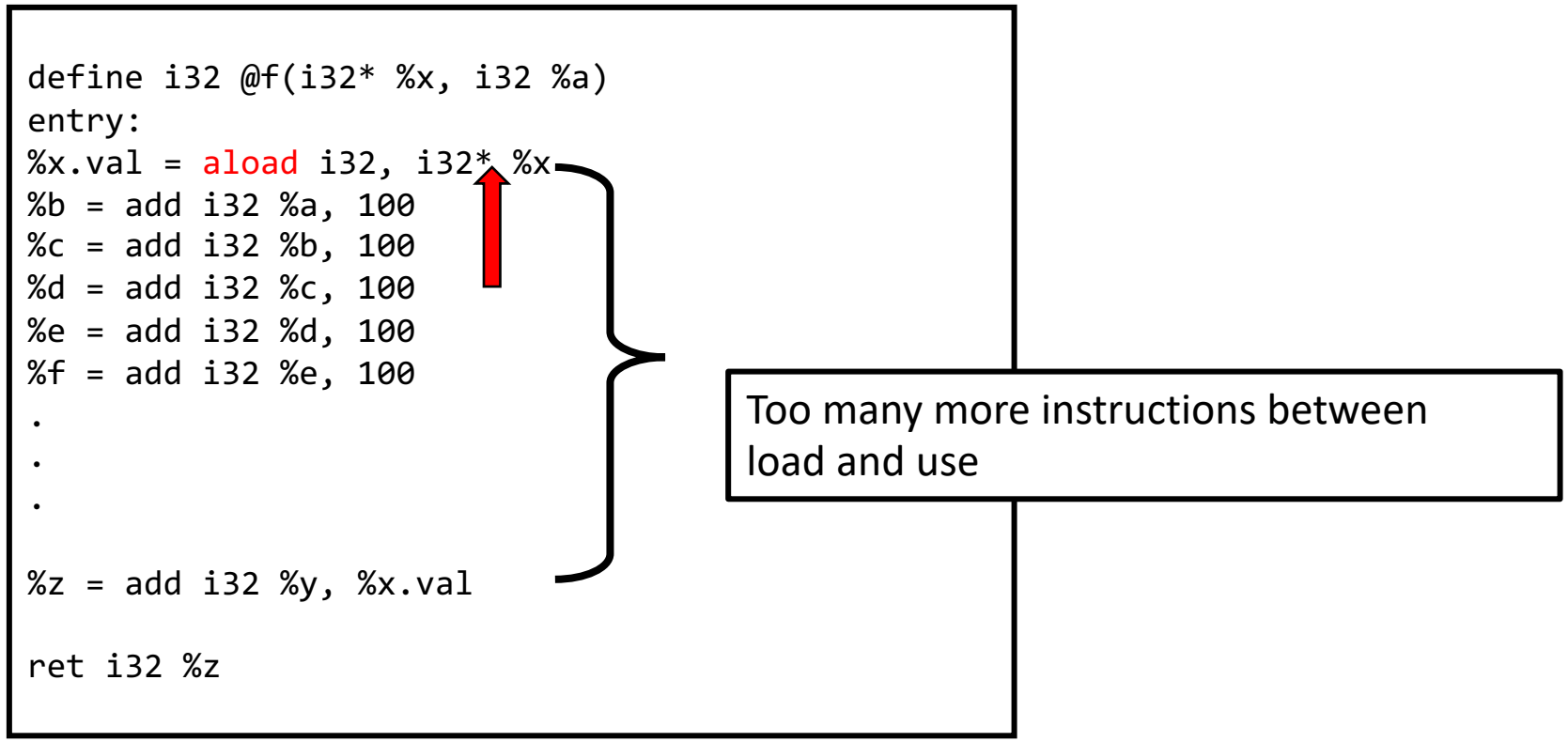
- aload로 cost 줄이기
  - load 하고 많은 시간 뒤에 사용되는 경우, aload로 효율 증가
  - 아예 최대한 load를 앞으로 빼고 aload로 바꾸기

```
define i32 @f(i32* %x, i32 %a)
entry:
%b = add i32 %a, 100
%c = add i32 %b, 100
%d = add i32 %c, 100
%x.val = load i32, i32* %x
%e = add i32 %d, 100
%f = add i32 %e, 100
.
.
.

%z = add i32 %y, %x.val

ret i32 %z
```



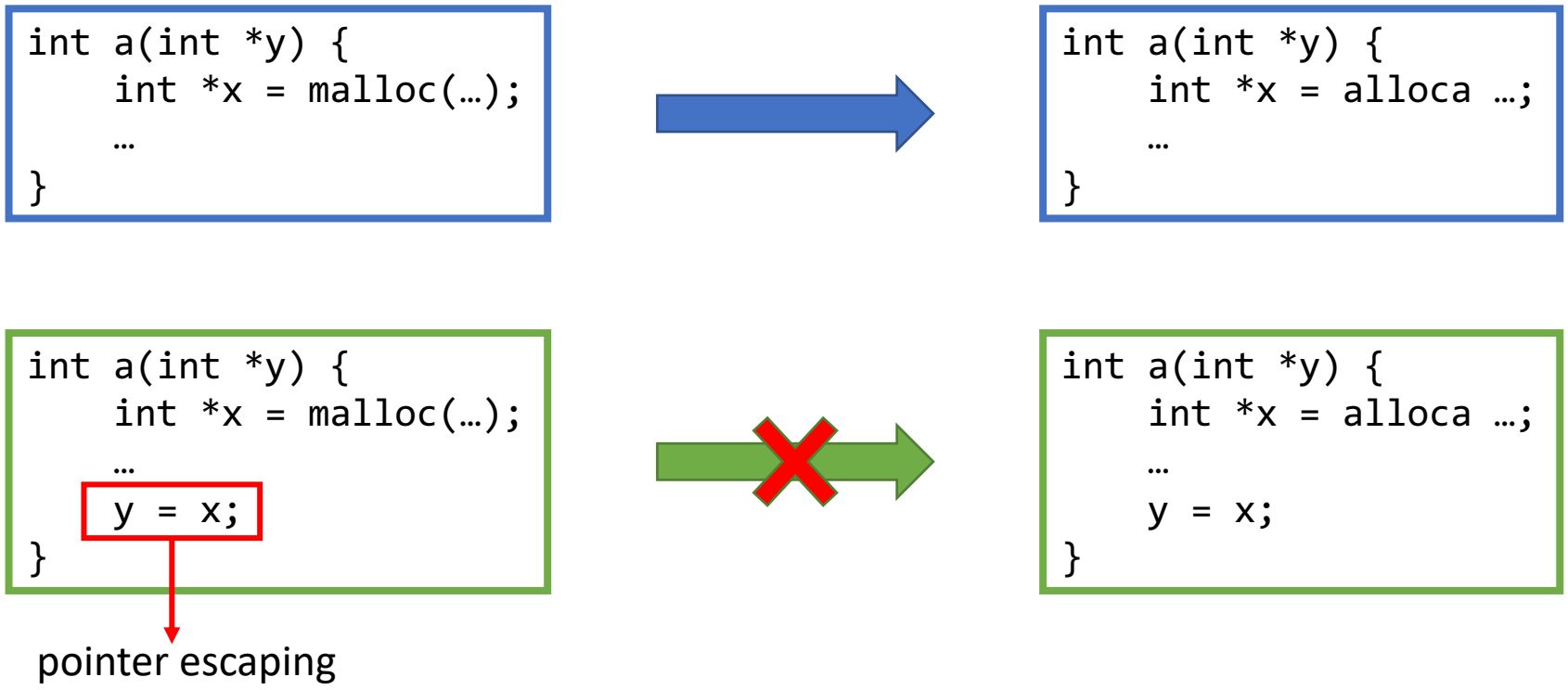
Too many instructions between load and use


define i32 @f(i32* %x, i32 %a)
entry:
%b = add i32 %a, 100
%c = add i32 %b, 100
%d = add i32 %c, 100
%x.val = aload i32, i32* %x
%e = add i32 %d, 100
%f = add i32 %e, 100
.
.
.
%z = add i32 %y, %x.val
ret i32 %z
Too many instructions between load and use


define i32 @f(i32* %x, i32 %a)
entry:
%x.val = aload i32, i32* %x
%b = add i32 %a, 100
%c = add i32 %b, 100
%d = add i32 %c, 100
%e = add i32 %d, 100
%f = add i32 %e, 100
.
.
.
%z = add i32 %y, %x.val
ret i32 %z
Too many more instructions between load and use

Memory access cost 최소화

- Malloc to alloca promoting



연산 cost 최소화

- branch cost 최소화
  - true_bb(cost=6)와 false_bb(cost=1) 교환
  - 현 basic block과 true_bb가 한 loop안에 있으면 true_bb와 false_bb를 교환하고 cond를 not cond로 교환

```
…
loop:
a = …
b = …
cond = icmp <condi> a b
br <cond> loop exit

exit:
…
```

```
…
loop:
a = …
b = …
cond = icmp <not condi> a b
br <cond> exit loop

exit:
…
```

연산 cost 최소화

- 불필요한 branch 제거
  - ternary operation으로 변환

```
…
br <cond> true_bb false_bb

true_bb:
x1 = …
br after

false_bb:
x2 = …
br after

after:
z = phi[x1, x2]
…
```

```
…
x1 = …
x2 = …
z = select <cond> x1 x2
…
```

연산 cost 최소화

- 다른 값싼 연산으로 치환
  - add -> mul, add -> sum, sum -> mul
  - add/sub 1~4 -> incr/decr
  - add/sub 0 -> mul 1
  - shl, lshr, ashr -> mul/div
  - and -> mul (i1인 경우)

```
x1 = add a, b
x2 = add x1, c
x3 = add x2, d
```

```
x3 = sum a, b, c, d
```

```
x1 = mul a, 2
x2 = mul b, 3
x3 = mul c, 3
x4 = sum x1, x2, x3
```

```
x4 = sum a, a, b, b, b, c, c, c
```

연산 cost 최소화

- 다른 값싼 연산으로 치환