




OPTIMIZATION IDEA OUTLINE

9조 남기태, 오진수, 이준영,한지훈



Team introduction- Swpp9

- 남기태 - 물교 19, 수업 초반에 이론으로 다루어 보았던 Github을 이용한 협업 과정에 참여함으로써 이에 익숙해지고 3번의 iteration 과정을 통해 체계적인 협업 계획의 중요성을 느껴보고 싶다. 추가로 처음 다루어 보는 컴파일러의 이해까지 얻어갈 수 있다면 아주 좋을 것 같다.
- 오진수 - 컴공 18, 소개원실 수업을 통해 단순한 코딩하는 연습이 아닌, 협업하는 과정, 코드 리뷰, github, CI에 익숙해지는 것 등을 배우고 싶다.
- 이준영- 컴공 18, 기존에 혼자 코딩으로 과제나 프로젝트를 해결하거나, 팀 프로젝트를 해도 단순히 코드 자체의 교환을 통해 진행했던 적이 많았는데 이번 수업을 통해서 github을 이용하여 팀원들과 소통하며 프로젝트를 진행하면서 함께 개발하는 것이 무엇인지에 대해 배우고 싶다.
- 한지훈- 컴공21, 이번 소개원실을 통해 혼자서 구상하고 코딩을 하는 과정을 넘어서 다른 사람들과 소통하고 일을 분담하며 개발해나가는 과정을 배우고자 합니다!

Optimization Objectives

- Improving code performance
- Reducing cost & memory usage
- Efficiency

Conditional branch optimization

- Branch true/false condition을 바꾼다.

Ex>while(condition)은 condition이 true일 때 branch_true<= cost가 크다!

```
1 ; optimization idea 1
2 ; reversing branch conditions and swap branch blocks
3
4 before:
5     %cond = icmp sgt i32 %r1, %r2
6     br i1 %cond, label %bb_true, label %bb_false
7
8 -----
9
10 after:
11     %cond = icmp sle i32 %r1, %r2
12     br i1 %cond, label %bb_false, label %bb_true
```

Constant folding & propagation

- Compile time 에 상수로 evaluate되는 값들에 모두 상수를 대입
- 한 쪽이 constant가 아니더라도 특수한 경우에 대해 compile time에 값을 미리 resolve할 수 있다.

```
; optimization idea 2
; replace instruction with constant literal
; iff the value of instruction is determined in compile time

before:
    %a = sub i32 5, 3
    %b = udiv i32 100, 20
    %c = mul i32 2, 4
    %d = shl i32 %c, %a
    %e = call i32 @f(i32 %b, i32 %d)

-----

after:
    ; %a = i32 2
    ; %b = i32 5
    ; %c = i32 8
    ; %d = i32 32
    %e = call i32 @f(i32 5, i32 32)
```

Arithmetic optimization

■ Cheaper instructions!

```
1  ; optimization idea 3
2  ; replace instructions with other cheaper instructions if possible
3  ; e.g. one of the operand is constant (or replaced with constant), add/sub in a multiples relationship
4
5  before:
6  %a = add i32 %r1, 3
7  %b = add i32 %r2, %r2
8  %c = lshr i32 %r3, 6
9  %d1 = add i32 %s0, %s1
10 %d2 = add i32 %d1, %s2
11 %d3 = add i32 %d2, %s3
12 %d4 = add i32 %d3, %s4
13 %d5 = add i32 %d4, %s5
14
15 -----
16
17 after:
18 %a1 = call i32 @incr_i32 (i32 %r1)
19 %a2 = call i32 @incr_i32 (i32 %a1)
20 %a = call i32 @incr_i32 (i32 %a2)
21 %b = mul i32 %r2, 2
22 %c = udiv i32 %r3, 64
23 %d5 = call i32 @sum_i32 (i32 %s0, i32 %s1, i32 %s2, i32 %s3, i32 %s4, i32 %s5)
24
```

Ternary optimization

- If/else: branch 2번 -> ternary 가 훨씬 싸다

```
; optimization idea 4
; replace phi with select when the value of instruction is constant (or replaced with constant)
; iff the branch has nothing to do with other basicblocks & functions

before:
    %cond = icmp eq i32 %r1, %r2
    br i1 %cond, label %bb_true, label %bb_false

bb_true: ; %before is the only pred
    %x1 = mul i32 1, 2
    br label %bb_end

bb_false: ; %before is the only pred
    %x2 = mul i32 3, 4
    br label %bb_end

bb_end:
    %x = phi i32 [%x1, %bb_true], [%x2, %bb_false]

-----

intermediate:
    %cond = icmp eq i32 %r1, %r2
    br i1 %cond, label %i_true, label %i_false

i_true:
    ; %x1 = mul i32 1, 2
    br label %i_end

i_false:
    ; %x2 = mul i32 3, 4
    br label %i_end

i_end:
    %x = phi i32 [i32 2, %i_true], [i32 12, %i_false]

-----

after:
    %cond = icmp eq i32 %r1, %r2
    %x = select i1 %cond, i32 2, i32 12
```

Aload optimization

- load가 되는 것과 쓰이는 것 사이의 cost 비교 -> aload 바꾸기
- Aload instuction을 최대한 위로 올리기

```
1 ; optimization idea 5
2 ; replace load with aload, then adjust the order of instruction
3 ; iff it can reduce the total cost and doesn't break dominations
4
5 define void @add(i32* %ptr1, i32* %ptr2, i32* %val) {
6     %v = load i32, i32* %val
7
8     %p1 = load i32, i32* %ptr1
9     %a1 = add i32 %v, %p1
10    store i32 %a1, i32* %ptr1
11
12    %p2 = load i32, i32* %ptr2
13    %a2 = add i32 %v, %p2
14    store i32 %a2, i32* %ptr2
15    ret void
16 }
17
18 -----
19
20 define void @add(i32* %ptr1, i32* %ptr2, i32* %val) {
21     %v = call i32 @aload_4(i32* %val)
22
23     %p1 = call i32 @aload_4(i32* %ptr1)
24     %p2 = call i32 @aload_4(i32* %ptr2)
25
26     %a1 = add i32 %v, %p1
27     store i32 %a1, i32* %ptr1
28     %a2 = add i32 %v, %p2
29     store i32 %a2, i32* %ptr2
30     ret void
31 }
```


Usage of Oracle

- Memory load/store가 많은 함수를 위주로 oracle로 대체했을 때 감소 폭이 가장 큰 함수의 이름을 oracle로 rename

```
; optimization idea 6
; choose a function to rename as @oracle
; with a internal evaluation algorithm, choose a function that can most reduce the total cost when it is renamed as @oracle
; e.g. doesn't have more than 50 IRs, replacing load wiht aload (idea 5) is not helpful, have a lot of arguments

before:
define void @cand1() {
    ...
}
define void @cand2() { ; renaming @cand2 as @oracle can most reduce the total cost
    ...
}
define void @cand3() {
    ...
}
define void @cand4() {
    ...
}

-----

after:
define void @cand1() {
    ...
}
define void @oracle() {
    ...
}
define void @cand3() {
    ...
}
define void @cand4() {
    ...
}
```

Dead code elimination

- Unreachable bb 를 찾아서 제거 (dfs등의 방법을 통해)
- Unused variable 제거



THANK YOU