Project Plan

소프트웨어 개발의 원리와 실습

10조

김진영 노종찬 신다연 이하린 진성현

Single Operation Replacement

- Use cheaper operations

```
example1:
    %and = and i1 %b1, %b2  ; 4 costs
    %xor = xor i1 %b1, %b2  ; 4 costs

    %two = add i64 %x, %x   ; 5 costs
    %neg = sub i64 0, %x    ; 5 costs
    %shl = shl i64 %x, 3    ; 4 costs
    %ashr = ashr i64 %x, 4  ; 4 costs
```



```
example1:
    %and = mul i1 %b1, %b2       ; 1 cost
    %xor = icmp ne i1 %b1, %b2   ; 1 cost

    %two = mul i64 %x, 2         ; 1 cost
    %neg = mul i64 %x, -1        ; 1 cost
    %shl = mul i64 %x, 8         ; 1 cost
    %ashr = sdiv i64 %x, 16      ; 1 cost
```

Single Operation Replacement

- Use cheaper operations

```
example2:
    ; y = x + 3
    %y = add i64 %x, 3    ; 5 costs


example3:
    ; nxti = i + 8
    ; %i is guaranteed to be multiple of 8
    %nxti = add i32 %i, 8    ; 5 costs
```

```
example2:
    ; y = x + 3
    %y.1 = call i64 @incr_i64(i64 %x)
    %y.2 = call i64 @incr_i64(i64 %y.1)
    %y   = call i64 @incr_i64(i64 %y.2)
    ; 3 costs in total


example3:
    ; nxti = i + 8
    ; %i is guaranteed to be multiple of 8
    %nxti.1 = sdiv i32 %i, 8
    %nxti.2 = call i32 @incr_i32(i32 %nxti.1)
    %nxti = mul i32 %nxti.2, 8
    ; 3 costs in total
```

Make Consecutive aloads

- Overlap aload waits

```
example4:
    ; %ptr1, %ptr2 point to stack area
    %val1 = load i64, i64* %ptr1
    %tmp1 = mul i64 %val1, %x

    %val2 = load i64, i64* %ptr2
    %tmp2 = mul i64 %val1, %val2
    ; 42 costs in total
```

```
example4:
    ; %ptr1, %ptr2 point to stack area
    %val1 = call i64 @aload_i64(i64* %ptr1)
    %val2 = call i64 @aload_i64(i64* %ptr2)

    %tmp1 = mul i64 %val1, %x
    %tmp2 = mul i64 %val1, %val2
    ; 26 costs in total
```

Branch Prediction

- Prefer false when looped
- Use domination relation to check loop

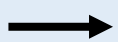
```
example5:
    ...
    %cmp = icmp slt i32 %nxti, %n
    br i1 %cmp, label %example5, label %next
```

```
example5:
    ...
    %cmp = icmp sge i32 %nxti, %n
    br i1 %cmp, label %next, label %example5
```

Minimize Heap Usage

- Use stack instead if possible
  - If the pointer is malloced and freed in one function

```
example6:
    %ptr = call i32* @malloc(i32 32)
    ...
    call void @free(i32* %ptr)
```



```
example6:
    %ptr = alloca i32, i32 32
    ...
```

Oracle

- Idea 1 : Choose existing function and change its name

- Idea 2 : Use as a multiple store function

```
define i64 @oracle(i64 val1, i64* ptr1, i64 val2, i64* ptr2):
begin:
    store i64 val1, i64* ptr1
    store i64 val2, i64* ptr2
```

Useful Existing Passes

- GVNPass (Global Value Numbering)
- SROAPass (Scalar Replacement of Aggregates)
- ADCEPass (Aggressive Dead Code Elimination)
- InlinerPass
- LoopUnrollPass