SWPP

Team 10

2024.04.25

팀원 소개

문진영 컴공 18

각오: cost를 늘리지 않겠습니다.

이서영 자전 20

각오: 화목한 팀플

조경원 컴공 20

각오: git, github 잘 써보고 싶습니다.

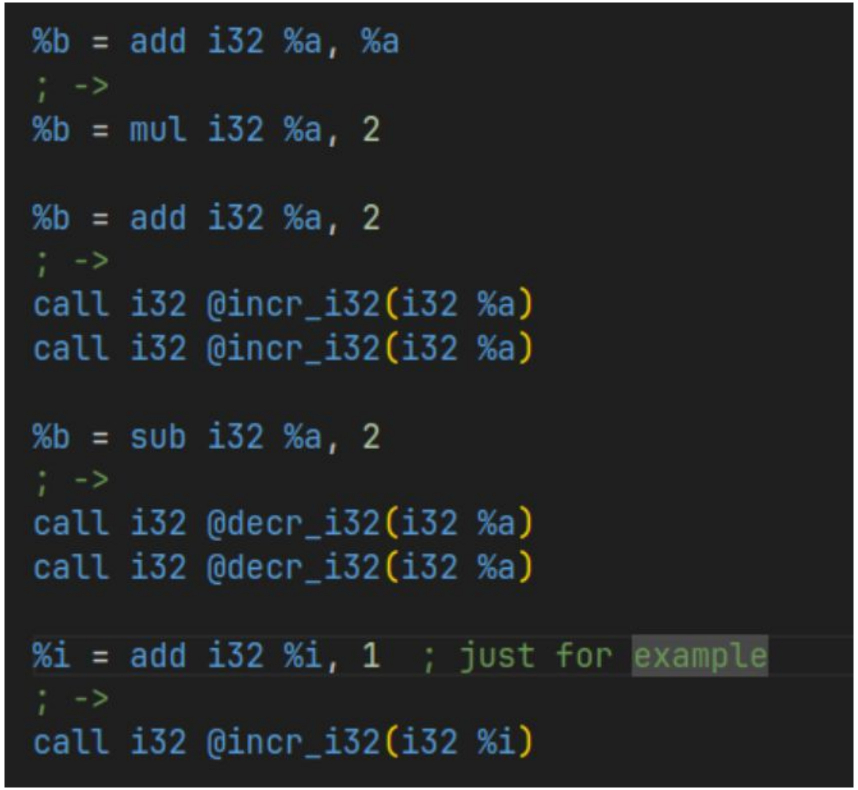
김재준 컴공 22

각오: LLVM IR과 친해지기

최적화 아이디어 카테고리

- Scalar operation
- Vector operation, Loop Unrolling with Array
- Control flow
- Function call
- Basic Block

Scalar operation: add/sub => mul, incr, decr



```
%b = add i32 %a, %a
; ->
%b = mul i32 %a, 2

%b = add i32 %a, 2
; ->
call i32 @incr_i32(i32 %a)
call i32 @incr_i32(i32 %a)

%b = sub i32 %a, 2
; ->
call i32 @decr_i32(i32 %a)
call i32 @decr_i32(i32 %a)

%i = add i32 %i, 1  ; just for example
; ->
call i32 @incr_i32(i32 %i)
```

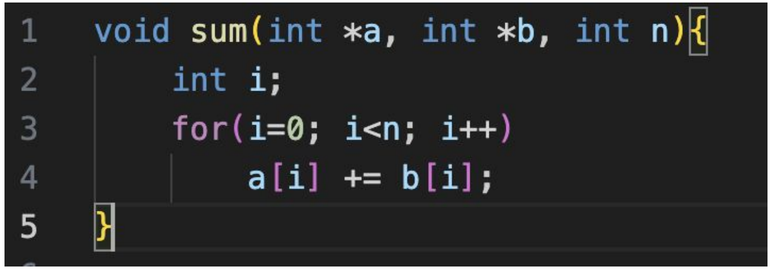
Pseudo Implementation Idea

1. Find instructions that performs add/sub operation &&
   One operand is ConstantInt
2. Check whether ConstantInt < 5

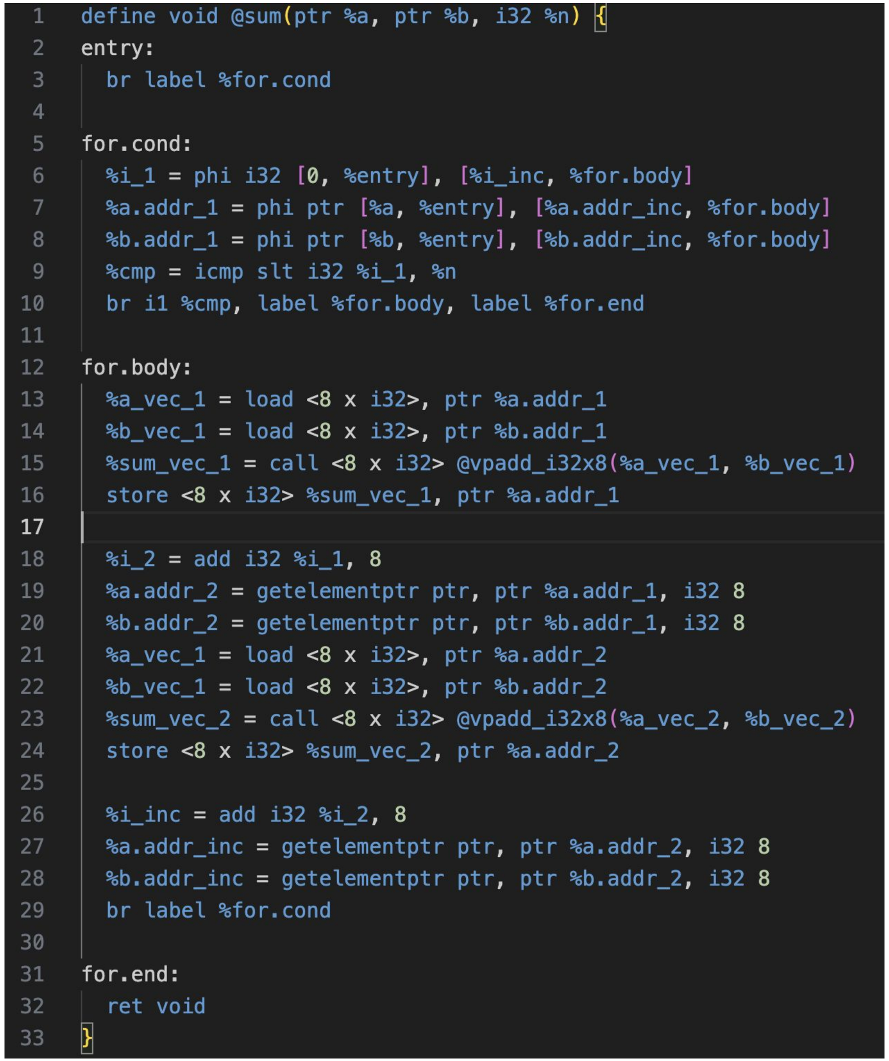
   add/sub operation cost: 5
   inc/dec operation cost: 1

3. If so, replace Instruction with ConstantInt times inc/dec

Vectorize & Loop Unrolling

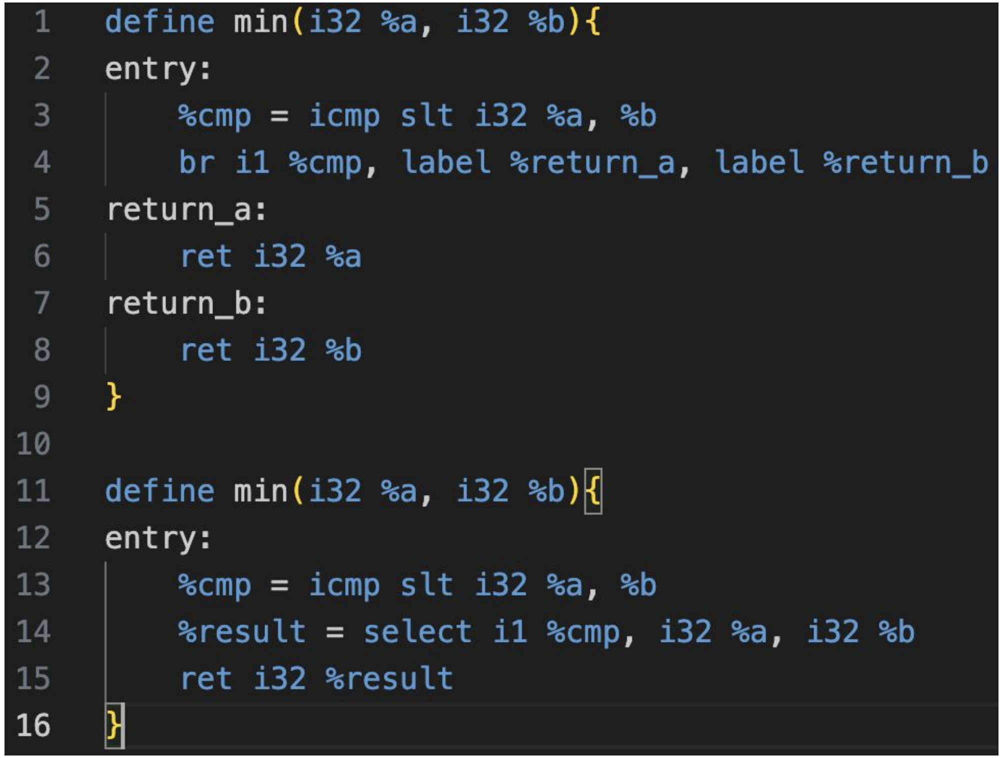


```c
void sum(int *a, int *b, int n){
    int i;
    for(i=0; i<n; i++)
        a[i] += b[i];
}
```



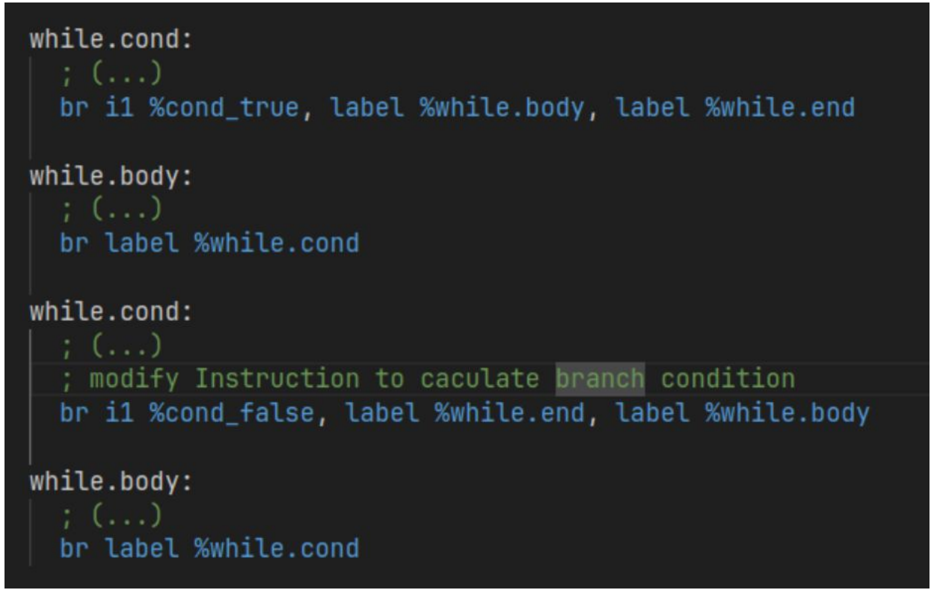
```llvm
define void @sum(ptr %a, ptr %b, i32 %n) {
entry:
  br label %for.cond

for.cond:
  %i_1 = phi i32 [0, %entry], [%i_inc, %for.body]
  %a.addr_1 = phi ptr [%a, %entry], [%a.addr_inc, %for.body]
  %b.addr_1 = phi ptr [%b, %entry], [%b.addr_inc, %for.body]
  %cmp = icmp slt i32 %i_1, %n
  br i1 %cmp, label %for.body, label %for.end

for.body:
  %a_vec_1 = load <8 x i32>, ptr %a.addr_1
  %b_vec_1 = load <8 x i32>, ptr %b.addr_1
  %sum_vec_1 = call <8 x i32> @vpadd_i32x8(%a_vec_1, %b_vec_1)
  store <8 x i32> %sum_vec_1, ptr %a.addr_1

  %i_2 = add i32 %i_1, 8
  %a.addr_2 = getelementptr ptr, ptr %a.addr_1, i32 8
  %b.addr_2 = getelementptr ptr, ptr %b.addr_1, i32 8
  %a_vec_1 = load <8 x i32>, ptr %a.addr_2
  %b_vec_1 = load <8 x i32>, ptr %b.addr_2
  %sum_vec_2 = call <8 x i32> @vpadd_i32x8(%a_vec_2, %b_vec_2)
  store <8 x i32> %sum_vec_2, ptr %a.addr_2

  %i_inc = add i32 %i_2, 8
  %a.addr_inc = getelementptr ptr, ptr %a.addr_2, i32 8
  %b.addr_inc = getelementptr ptr, ptr %b.addr_2, i32 8
  br label %for.cond

for.end:
  ret void
}
```

Ternary Operation



```
define min(i32 %a, i32 %b){
entry:
    %cmp = icmp slt i32 %a, %b
    br i1 %cmp, label %return_a, label %return_b
return_a:
    ret i32 %a
return_b:
    ret i32 %b
}

define min(i32 %a, i32 %b){
entry:
    %cmp = icmp slt i32 %a, %b
    %result = select i1 %cmp, i32 %a, i32 %b
    ret i32 %result
}
```

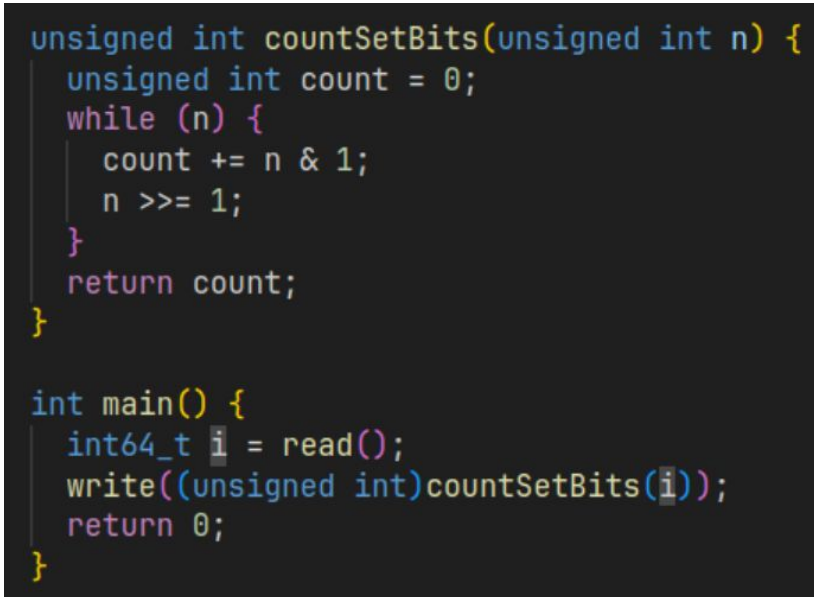
Control flow: (Ternary Operation) + Negative loop condition



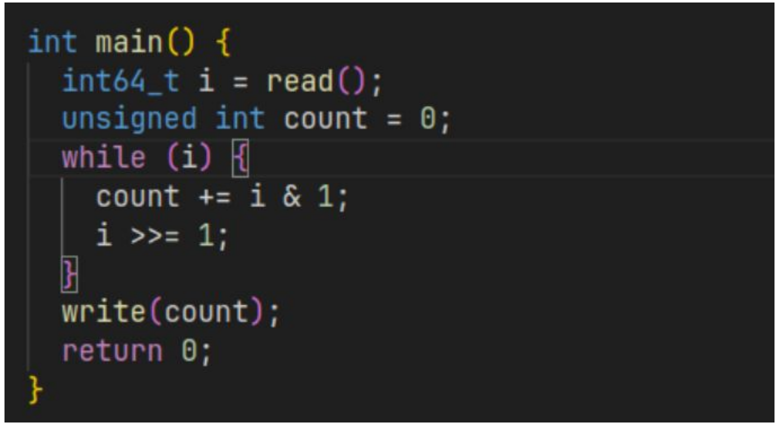
```
while.cond:
  ; (...)
  br i1 %cond_true, label %while.body, label %while.end

while.body:
  ; (...)
  br label %while.cond

while.cond:
  ; (...)
  ; modify Instruction to caculate branch condition
  br i1 %cond_false, label %while.end, label %while.body

while.body:
  ; (...)
  br label %while.cond
```

true_bb cost > false_bb cost 이므로, branch condition이 true 일 때
loop body가 아니라 loop end로 가도록 transform

Function inlining



```c
unsigned int countSetBits(unsigned int n) {
  unsigned int count = 0;
  while (n) {
    count += n & 1;
    n >>= 1;
  }
  return count;
}

int main() {
  int64_t i = read();
  write((unsigned int)countSetBits(i));
  return 0;
}
```



```c
int main() {
  int64_t i = read();
  unsigned int count = 0;
  while (i) {
    count += i & 1;
    i >>= 1;
  }
  write(count);
  return 0;
}
```

Pseudo Implementation Idea

1. Find Callee Function which is not recursive.
2. Find Caller and replace call instruction with Callee Function implementation
3. Modify args

Recursive call

```
while (1) {
    uint64_t curr_data = *curr;
    uint64_t *next;
    if (curr_data > data) {
        …
        If (next == NULL) {
            …
            return 1;
        }
        curr = next;
        continue;
    } else if (curr_data < data) {
        …
        If (next == NULL) {
            …
            return 1;
        }
        curr = next;
        continue;
    } else
    return 0;
 }
}
```

```
func(uint64 *curr) {
    uint64 curr_data = *curr;
    uint64_t *next;
    if (curr_data > data) {
        …
        If (next == NULL) {
            …
            return 1;
        }
        curr = next;
        func(curr, next); //rcall
    } else if (curr_data < data) {
        …
        If (next == NULL) {
            …
            return 1;
        }
        curr = next;
        func(curr, next); //rcall
    } else
    return 0;
 }
}
```

Merge Blocks of unconditional branch

```
BB1:
        …(BB1)
        Br BB2
BB2:
        …(BB2)
```

```
BB1:
        …(BB1)
        …(BB2)
BB2:
        …
```

Pseudo Implementation Idea

1. Find the destination block.
2. Copy and paste instructions of the destination.