Zad 2.

sobota, 3 czerwca 2023 18:49

Zadanie 2. Ile razy zostanie zawołana funkcja «my_strlen» w funkcji «my_index» i dlaczego? Dodaj atrybut[2] «pure» do funkcji «my_strlen». Czemu tym razem kompilator był w stanie lepiej zoptymalizować funkcję «my_index»? Czym charakteryzują się **czyste funkcje**? Następnie uzupełnij ciało funkcji «my_strlen» tak, by wykonywała to samo co «strlen». Następnie usuń atrybut «pure» i dodając słowo kluczowe «static» zawęź zakres widoczności funkcji do bieżącej jednostki translacji. Co się stało w wyniku przeprowadzenia **inliningu**? Czy kompilatorowi udało się samemu wywnioskować, że funkcja jest czysta?

```
1 __attribute__((leaf))
2   size_t my_strlen(const char *s);
3
4 const char *my_index(const char *s, char v) {
5   for (size_t i = 0; i < my_strlen(s); i++)
6     if (s[i] == v)
7       return &s[i];
8   return 0;
9 }
```

my_strlen(s) wywoła się tyle razy, ile znaków ma s (można nawet w godbolcie sprawdzić). Dzieje się tak, bo kompilator nie wie, czy my_strlen(s) nie powoduje efektów ubocznych (szczególnie takich zmieniających wartość pod s)

czyste funkcje charakteryzują się brakiem efektów ubocznych, zatem kompilator może już coś poddymić

```
my_index:
        pushq   %r12
        movl    %esi, %r12d
        pushq   %rbp
        xorl    %ebp, %ebp
        pushq   %rbx
        movq    %rdi, %rbx
.L2:
        movq    %rbx, %rdi
        call    my_strlen
        cmpq    %rax, %rbp
        jnb     .L8
        cmpb    %r12b, (%rbx,%rbp)
        jne     .L3
        leaq    (%rbx,%rbp), %rax
        jmp     .L1
.L3:
        incq    %rbp
        jmp     .L2
.L8:
        xorl    %eax, %eax
.L1:
        popq    %rbx
        popq    %rbp
        popq    %r12
        ret
```

```
my_index:
        pushq   %rbp
        movl    %esi, %ebp
        pushq   %rbx
        movq    %rdi, %rbx
        pushq   %rcx
        call    my_strlen
        movq    %rax, %rdx
        movq    %rbx, %rax
        addq    %rbx, %rdx
.L2:
        cmpq    %rdx, %rax
        je      .L8
        leaq    1(%rax), %rcx
        cmpb    %bpl, (%rax)
        je      .L1
        movq    %rcx, %rax
        jmp     .L2
.L8:
        xorl    %eax, %eax
.L1:
        popq    %rdx
        popq    %rbx
        popq    %rbp
        ret
```

+ pure

```
my_index:
        pushq   %rbp
        movl    %esi, %ebp
        pushq   %rbx
        movq    %rdi, %rbx
        pushq   %rcx
        call    strlen
        xorl    %edx, %edx
.L2:
        cmpq    %rax, %rdx
        jnb     .L8
        cmpb    %bpl, (%rbx,%rdx)
        jne     .L3
        leaq    (%rbx,%rdx), %rax
        jmp     .L1
.L3:
        incq    %rdx
        jmp     .L2
.L8:
        xorl    %eax, %eax
.L1:
        popq    %rdx
        popq    %rbx
        popq    %rbp
        ret
```

+ static
- pure

```
ret
```

```
ret
```

+ pure

```
ret
```

+ static
- pure

udało mu się nawet ogarnąć co funkcja robi, ale nie zinline'owało