Problem 1

Original:

```
class Foo {
    public int maxv(int x, int y) {
        if (x < y) return x;
        else if (y < x)
            return this.maxv(y, x);
        else
            return x;
    }
}
```

Translated:

```
; arguments for maxv are: this pointer in %rdi, int x in %rsi, int y in %rdx
maxv:
    push %rbp           ; push old frame pointer onto stack
    mov %rsp, %rbp      ; save current frame pointer into %rbp
    cmp %rsi, $rdx      ; compare x and y
    jl .return_x        ; if x < y, return x
    jg .recurse         ; if y > x, return y
    je .return_x        ; if x = y, return x

.return_x:
    mov %rsi, %rax      ; move x into return register
    pop %rbp            ; pop frame pointer from stack
    ret

.recurse:
    mov 0(%rdi), %rax   ; load vtable pointer into %rax
    mov 8(%rax), %rax   ; load method pointer from vtable
    mov %rsi, %rcx      ; move x into temp register
    mov %rdx, %rsi      ; move y into first arg spot
    mov %rcx, %rdx      ; move x from temp into second arg spot
    call %rax           ; recursively call maxv
    pop %rbp            ; pop frame pointer from stack
    ret
```

Problem 2

Original:

```
class Base {
    int a;
    int b;
    public int f(int n) {
        b = n + 1;
        return n + 2;
    }
    public int g(int n) {
        return a + n;
    }

    public int setA(int v) {
        a = v;
        return a;
    }

    public int setB(int v) {
        b = v;
        return b;
    }
}

class Sub extends Base {
    int c;
    public int setC(int v) {
        c = v;
        return c;
    }

    public int g(int n) {
        c = this.f(b);
        return b + n;
    }
}
```

Translated:

```
; arguments for g are: this pointer in %rdi, int n in %rsi
g:
    push %rbp           ; push old frame pointer onto stack
    mov %rsp, %rbp      ; save current frame pointer into %rbp

    push %rsi           ; save `n` onto stack

    mov 16(%rdi), %rsi  ; load `b` into %rsi (assumes that `b` is at offset 16
                        ; in Base class: vtable pointer at offset 0, `a` is at
                        ; offset 8) before calling `f`

    mov 0(%rdi), %rax   ; load vtable pointer into %rax

    call 8(%rax)        ; call `f` (assumes that f is first function in vtable)
```

```
    mov %rax, 24(%rdi)  ; store return value of `f` in `c` (assumes that `c`
                        ; is at offset 24 in Sub class: vtable pointer at
                        ; offset 0, `a` at offset 8, `b` at offset 16)

    pop %rsi            ; load `n` from stack

    mov 16(%rdi), %rax  ; load `b` into return register
    add %rsi, %rax      ; add `n` to `b`

    pop %rbp            ; pop old frame pointer from stack
    ret
```