컴파일러개론 Assign 05-2

과제 제출일자 : 2019/11/25
학번 : 201502085, 이규봉
분반 : 01
과제 : C언어 -> 자바 바이트 코드 컴파일러 작성

문제 해결 방법

MiniC.g4의 각 문장에 해당하는 부분의 enter, exit 메서드에 해당하는 내용을 작성합니다. 기본적으로 제가 작성한 코드는, enter에서 expr의 내용에 해당하는 노드들에 대해 newTexts에 저장해 놓고 exit 메서드에서 해당 텍스트들을 구문 구조에 따라 배치하는 식으로 동작합니다. 또한, exit 메서드는 MiniC.g4의 각 문장들의 구문 구조의 리프 노드에서 (터미널 노드) 위로 거슬러 올라가며 해당 구문의 newTexts에 저장된 문장을 다시 newTexts에 저장합니다. 최종적으로 exitProgram의 실행 시에 newTexts에 저장된 문장들을 전부 출력하면 test.c의 모든 문장들이 해당하는 바이트 코드로 바꿔 컴파일되게 됩니다.

상세한 코드의 작동 방법은 코드의 각 메서드들에 주석으로 설명하였습니다.

스크린샷 및 코드 실행 사진

Test 1

```
int add(int x, int y) {
    int z ;
    z = x+y;
    return z;
}

void main () {
    int a = 1;
    int b = 0;

    if(a or b){
        _print(add(--a, ++b));
    }

}
```

```
C:₩Compiler-HW>jasmin.jar Test.j Test.class

C:₩Compiler-HW>java Test
1

C:₩Compiler-HW>
```

```
.class public Test
.super java/lang/Object
; strandard initializer
.method public <init>()V
aload_0
invokenonvirtual java/lang/Object/<init>()V
return
.end method
.method public static add(II)I
.limit stack 32
.limit locals 32
iload_0
iload_1
iadd
istore_2
iload_2
ireturn
.end method
.method public static main([Ljava/lang/String;)V
.limit stack 32
.limit locals 32
ldc 1
istore_1
ldc 0
istore_2
iload_1
iload_2
ifeq    label1
pop
ldc 1
label1:
ifeq    label8
getstatic java/lang/System/out Ljava/io/PrintStream;
iload_1
ldc 1
isub
iload_2
ldc 1
iadd
invokestatic Test/add(II)I
```

```
invokevirtual java/io/PrintStream/println(I)V
label8:
return
.end method
```

Test 2

```
int add(int x, int y) {
    int z ;
    z = x+y;
    return z;
}

int sub(int x, int y){
    int z;
    z = x-y;
    return z;
}

int div(int x, int y){
    int z;
    z = x / y;
    return z;
}

int mul(int x, int y){
    int z;
    z = x * y;
    return z;
}

void main () {
    int a = 1;
    int b = 0;

    a = div(6, 2);

    if(a or b){
        _print(mul(add(--a, ++b), 3));
    }

}
```

```
C:₩Compiler-HW>jasmin.jar Test.j Test.class

C:₩Compiler-HW>java Test
9

C:₩Compiler-HW>
```

```
.class public Test
.super java/lang/Object
; strandard initializer
.method public <init>()V
aload_0
invokenonvirtual java/lang/Object/<init>()V
return
.end method
.method public static add(II)I
.limit stack 32
.limit locals 32
iload_0
iload_1
iadd
istore_2
iload_2
ireturn
.end method
.method public static sub(II)I
.limit stack 32
.limit locals 32
iload_0
iload_1
isub
istore_2
```

```
iload_2
ireturn
.end method
.method public static div(II)I
.limit stack 32
.limit locals 32
iload_0
iload_1
idiv
istore_2
iload_2
ireturn
.end method
.method public static mul(II)I
.limit stack 32
.limit locals 32
iload_0
iload_1
imul
istore_2
iload_2
ireturn
.end method
.method public static main([Ljava/lang/String;)V
.limit stack 32
.limit locals 32
ldc 1
istore_1
ldc 0
istore_2
ldc 6
ldc 2
invokestatic Test/div(II)I
istore_1
iload_1
iload_2
ifeq    label1
pop
ldc 1
label1:
ifeq    label8
getstatic java/lang/System/out Ljava/io/PrintStream;
iload_1
ldc 1
isub
iload_2
ldc 1
iadd
invokestatic Test/add(II)I
ldc 3
invokestatic Test/mul(II)I
invokevirtual java/io/PrintStream/println(I)V
label8:
return
.end method
```

Test 3

```
1  void main () {
2      int a = 4;
3
4      while(a > 0){
5          a = a - 1;
6          _print(a);
7      }
8
9  }
10
```

```
C:₩Compiler-HW>jasmin.jar Test.j Test.class

C:₩Compiler-HW>java Test
3
2
1
0

C:₩Compiler-HW>
```

```
.class public Test
.super java/lang/Object
; standard initializer
.method public <init>()V
aload_0
invokenonvirtual java/lang/Object/<init>()V
return
.end method
.method public static main([Ljava/lang/String;)V
.limit stack 32
.limit locals 32
ldc 4
istore_1
label5:
iload_1
ldc 0
isub
ifgt    label0
ldc 0
goto    label1
label0:
ldc 1
label1:
ifeq    label4
iload_1
ldc 1
isub
istore_1
getstatic java/lang/System/out Ljava/io/PrintStream;
iload_1
invokevirtual java/io/PrintStream/println(I)V
goto    label5
label4:
return
.end method
```

이상으로, 주어진 c 코드에 대해 이상 없이 j 코드로 컴파일되는 것을 확인했으며, jasmin을 통해 class 파일로 변환 후 정상적으로 작동하는 것 까지 확인하였습니다.

총 과제 수행에 걸린 시간

자바 바이트 코드가 익숙하지 않아 코드 분석에 다소 많은 시간이 소요되었습니다. 3 ~ 4일 정도 걸린 것 같습니다.