Phase 4 : Code Generation

Program.java

```
public void cgen(PrintWriter s) {

                CgenClassTable cct = new

    CgenClassTable(getClasses(), s);

}
```

codeGeneration.CgenClassTable.java

The following steps are performed in the constructor

```
try {
    installBasicClasses();
    installClasses(cls);
    buildInheritanceTree();
    code();
    exitScope();
} catch (NoScopePresentException e) {
    Utilities.fatalError(e);
    }
```

Generating code for classes

- *installClasses(cls)* would generate the code for all the classes by iterating through all the classes.
- code() performs all the data related functions, and the in the last line calls *codeMethods()*
- codeMethods() functioncalls the treeNodes.Method.cgen(…) function, which needs to be implemented by you.

treeNodes.Method.cgen(…)

- This might look familiar to you now:

```
public void cgen(java.io.PrintWriter str,
    codeGeneration.CgenNode cls ) {
        cls.setMethodFormals(formals);
        expr.cgen(str, cls);
    }
```

- Implement all cgen(java.io.PrintWriter str, codeGeneration.CgenNode cls) methods in treeNodes.* classes

Tips:

- Look at spim.pdf if you would like to look at the instrction set.
- Compare the code generated by the binary and use diff to find places where you might be going wrong.