Génération de code trois adresses en Java

Alexis Nasr
Franck Dary
Pacôme Perrotin

Compilation – L3 Informatique
Département Informatique et Interactions
Aix Marseille Université

Génération du code trois adresses en Java

- La génération du code trois addresses est réalisée lors d'un parcours de l'arbre abstrait.
- Ce parcours est réalisé à l'aide d'une classe visiteur, qui étend `SaDepthFirstVisitor`.
- Les classes implémentant les instructions et les opérandes se trouvent dans le package `c3a`.
- Les instructions générées sont stockées dans une structure linéaire indicée.

Le package c3a

Il est composé de 25 classes se décomposant de la façon suivante :

- 17 classes correspondant à des instructions
- 6 classes correspondant à des opérandes
- 1 visiteur
- 1 classe principale

Les instructions

- Une classe abstraite `C3aInst`
- 16 classes héritant de `C3aInst`
  - `C3aInstAdd`
  - `C3aInstMult`
  - `C3aInstSub`
  - `C3aInstDiv`
  - `C3aInstJump`
  - `C3aInstJumpIfLess`
  - `C3aInstJumpIfEqual`
  - `C3aInstJumpIfNotEqual`
  - `C3aInstAffect`
  - `C3aInstCall`
  - `C3aInstParam`
  - `C3aInstReturn`
  - `C3aInstFBegin`
  - `C3aInstFEnd`
  - `C3aInstRead`
  - `C3aInstWrite`

Exemple : C3aInstAdd

```
public class C3aInstAdd extends C3aInst{
  public C3aOperand op1;    // opérande 1
  public C3aOperand op2;    // opérande 2
  public C3aOperand result; // résultat

public C3aInstAdd(C3aOperand op1, C3aOperand op2,
                  C3aOperand result, String comment){...}

public String toString(){...}

public <T> T accept(C3aVisitor <T> visitor) {
  return visitor.visit(this);
}
}
```

Les opérandes

- Une classe abstraite : `C3aOperand`
- 5 classes héritant de `C3aOperand` :
  - `C3aConstant`
  - `C3aFunction`
  - `C3aLabel`
  - `C3aTemp`
  - `C3aVar`

C3aConstant

```
public class C3aConstant extends C3aOperand{
    public int val;
    public C3aConstant(int val){ ... }
    public String toString(){ ... }
    public <T> T accept(C3aVisitor <T> visitor) { ... }
}
```

C3aVar

```
public class C3aVar extends C3aOperand{
    public TsItemVar item;
    public C3aOperand index;

    public C3aVar(TsItemVar item, C3aOperand index){...}
    public String toString(){...}

    public <T> T accept(C3aVisitor <T> visitor) {...}
}
```

C3aTemp

```
public class C3aTemp extends C3aOperand{
    public int num;
    public C3aTemp(int num){...}
    public String toString(){ ... }
    public <T> T accept(C3aVisitor <T> visitor) { ... }
}
```

C3aFunction

```
public class C3aFunction extends C3aOperand{
    public TsItemFct val;
    public C3aFunction(TsItemFct val){ ... }
    public String toString(){ ... }
    public <T> T accept(C3aVisitor <T> visitor) { ... }
}
```

C3aLabel

```
public class C3aLabel extends C3aOperand{
    public int number; // numéro de l'étiquette
    private int line;   // ligne de l'opération correspondant à
    public C3aLabel(int number){ ... }
    public String toString(){ ... }
    public <T> T accept(C3aVisitor <T> visitor) { ... }
}
```

La classe générale C3a

```
public class C3a{
    public List<C3aInst> listeInst;
    // compteur des temporaires, pour générer des noms uniques
    private int tempCounter;
    // étiquette de la prochaine instruction,
    // on retarde l'ajout de l'étiquette
    private C3aLabel nextLabel;
    private int labelCounter;
    public C3aConstant True; // constantes utilisées partout
    public C3aConstant False;// constantes utilisées partout

    public C3a(){
        this.listeInst = new ArrayList<C3aInst>();
        this.labelCounter = 0;
        this.tempCounter = 0;
        this.nextLabel = null;
        this.True = new C3aConstant(1);
        this.False = new C3aConstant(0);
    }
}
```

Interface de C3a

```
public void ajouteInst(C3aInst inst){ ...}
    if(this.nextLabel != null){
        inst.setLabel(this.nextLabel);
        this.nextLabel = null;
    }
    this.listeInst.add(inst);
}
public C3aLabel newAutoLabel(){
    return new C3aLabel(this.labelCounter++);
}
public C3aTemp newTemp(){
    return new C3aTemp(this.tempCounter++);
}

public void addLabelToNextInst(C3aLabel label){
    this.nextLabel = label;
}
```

Parcours de l'arbre abstrait et production du code : Sa2c3a

```
public class Sa2c3a extends SaDepthFirstVisitor <C3aOperand> {
    private C3a c3a;

    public Sa2c3a(SaNode root){
        c3a = new C3a();
        root.accept(this);
    }

    public C3aOperand visit(SaProg node){ ... }
    public C3aOperand visit(SaDecTab node){ ... }
    public C3aOperand visit(SaDecFonc node) { ... }
    ...
}
```

Intégration dans le compilateur

```
System.out.print("[BUILD SA] ");
Sc2sa sc2sa = new Sc2sa();
tree.apply(sc2sa);
SaNode saRoot = sc2sa.getRoot();

System.out.print("[BUILD TS] ");
Ts table = new Sa2ts(saRoot).getTableGlobale();

System.out.println("[BUILD C3A]");
C3a c3a = new Sa2c3a(saRoot, table).getC3a();

System.out.println("[PRINT C3A]");
c3a.affiche(baseName);
```

Le visiteur C3aVisitor

```
public interface C3aVisitor <T> {
    public T visit(C3aInstAdd inst);
    public T visit(C3aInstCall inst);
    public T visit(C3aInstFBegin inst);
    ...
    public T visit(C3aConstant oper);
    public T visit(C3aLabel oper);
    public T visit(C3aTemp oper);
    public T visit(C3aVar oper);
    public T visit(C3aFunction oper);
}
```