Compilateur Deca : Manuel Utilisateur

Équipe gl58

January 23, 2017

1 Introduction au compilateur

Le compilateur Decac prend en argument un fichier source en Deca et le compile pour donner un fichier en code assembleur.
Ce compilateur gère toutes les étapes de la compilation, c'est-a-dire le parsing (analyse lexical et syntaxique) du fichier source, l'analyse contextuel du code et la génération du code assembleur.

2 Exécution de decac

Pour lancer le compilateur, il faut utiliser la commande `decac`. Cette commande possède différentes options d'exécution:

- `decac`
  Affiche les options possibles.
- `decac -b`
  Affiche une bannière de l'équipe qui a programmé ce compilateur.
- `decac -p <ficher deca>`
  Parse le fichier passé en argument et affiche sa decompilation.
- `decac -v <fichier deca>`
  Vérifie contextuellement le fichier deca. Affiche une erreur si il y en a.
- `decac -n <fichier deca>`
  Exécute le fichier deca en ignorant les erreurs de débordement à l'exécution.
- `decac -r X <fichier deca>`
  Execute le fichier deca en limitant les registres disponible à `R{0}...R{X}`.
- `decac -d <fichier deca>`
  Active les traces de debug lors de l'exécution.
- `decac -P <fichier(s) deca>`
  Si il y a plusieurs fichiers sources, les exécutent en parallèle.

Plusieurs options peuvent être appelées simultanément, `-p` et `-v` exclus. Le fichier `.ass` obtenu en sorti du compilateur peut être passé en exécutable avec la commande `ima <fichier assembleur>`.

3 Erreurs levées par le compilateur

3.1 Erreurs lexicales

3.2 Erreurs syntaxiques

3.3 Erreurs contextuelles

Voici les erreurs qui peuvent être levées par l'analyse contextuelle :

- `type or class undefined`
  Est levée si un identifier de type est attendu, mais l'identifier trouvé n'est pas un type prédéfini ou une classe qui a été définie.

- `class extends type`
  Est levée si, lors de la déclaration d'une classe, l'identifier suivant `extends` est un type (`int,float,boolean,void`);

- `class already defined`
  Est levée si une même classe est déclarée plusieurs fois.

- `field already defined`
  Est levée si un champ est déclaré plusieurs fois ou déclaré avec le même nom qu'une méthode existante.

- `field cannot be void`
  Est levée si un champ a pour type `void`.

- `method already defined`
  Est levée si une méthode est déclarée plusieurs fois dans la même classe ou déclarée avec le même nom qu'un champ existant.

- `method overrides method with different signature`
  Est levée si une méthode essaie d'écraser une méthode avec une différente signature (différent type, différent nombre de paramètres ou différent types de paramètres).

- `parameter cannot be void`
  Est levée si un paramètre d'une méthode a pour type `void`.

- `parameter already defined`
  Est levée si plusieurs paramètres d'une même méthode ont le même nom.

- `variable already defined`
  Est levée si une variable est déclarée plusieurs fois.

- `variable cannot be void`
  Est levée si une variable a pour type `void`.

- `return called in void method`
  Est levée si `return` est appelée dans une méthode qui a pour type `void`.

- `type of expression must match method type`
  Est levée si l'expression renvoyée n'est pas du même type que la méthode.

- `expected type found class`
  Est levée si l'expression est une instance de classe alors qu'une expression de type `int,float` ou `boolean` est attendue.

- `incompatible class type`
  Est levée si la classe attendue n'est pas une super-classe de la classe renvoyé. Ce cas est le seul qui permet d'initier l'instance d'une classe avec une différente classe.

- `expected different type`
  Est levée si l'expression trouvée n'est pas du type attendu, sauf si l'expression est de type `int` et le type attendu est `float`. Dans ce dernier cas, on converti l'`int` en `float`.

- `only string, int and float expressions can be printed`
  Est levée si un `void`,`null` ou une instance de classe est mis en paramètre de `print` ou `println`.

- `condition must be boolean`
  Est levée si la condition dans un `if` ou `while` n'est pas booléen.

- `operands must be int or float`
  Est levée si les opérandes d'un opération arithmétique (ou d'une comparaison) ne sont pas numériques.

- `operand must be int or float`
  Est levée si l'opérande du moins unaire n'est pas numérique.

- `operands must be boolean`
  Est levée si les opérandes d'une opération booléenne n'est pas `boolean`.

- `operand must be boolean`
  Est levée si l'opérande de `Not(!)` n'est pas booléen.

- `operands must have same type`
  Est levée si les operandes d'une comparaison n'ont pas le meme type. Notamment, on ne peut pas comparer un `int` et un `float`.

- `incompatible types for cast`
  Est levée si expression est `Cast` dans un type incompatible a son propre type. Le seul `Cast` de type accepté est le passage de `int` à `float`.

- `type cast to class`
  Est levée si une expression de type `int,float` ou `boolean` est `Cast` dans une `class`.

- `class cast to type`
  Est levée si l'instance d'une classe est `Cast` en `int,float` ou `boolean`.

- `classes incompatible for cast`
  Est levée si l'instance d'une classe est `Cast` dans une classe incompatible. Cela se produit quand les deux classes ne sont pas dans la même hiérarchie de classes.

- `identifier is not a class`
  Est levée si l'identifier suivant `new` n'est pas un nom de classe.

- `cannot call this in main`
  Est levée si `this` est appelé en dehors d'une déclaration de classe.

- `left operand not instance of a class`
  Est levée si l'identifier a gauche d'un appel de champ n'est pas l'instance d'une classe.

- `no such field in class`
  Est levée si l'identifier à droite d'un appel de champ ne correspond pas à un champ de la classe à gauche.

- `identifier is not a field`
  Est levée si l'identifier à droite d'un appel de champ correspond à une méthode.

- `field is protected`
  Est levée si on essaie d'appeler un champ `protected` en dehors de la classe à laquelle il appartient.

- `expression not instance of a class`
  Est levée si l'identifier a gauche d'un appel de méthode n'est pas l'instance d'une classe.

- `direct method call in main`
  Est levée si une méthode est appelée sans préciser la classe, en dehors d'une déclaration de classe.

- `identifier is not a method`
  Est levée si l'identifier a droite d'un appel de méthode correspond a un champ.

- `number of parameters does not match signature`
  Est levée si une méthode est appelée avec un nombre de paramètres différent du nombre de paramètre dans sa signature.

- `parameter type does not match signature`
  Est levée si le type d'un paramètre dans un appel de méthode ne correspond pas au type de paramètre dans la signature de la méthode.

- `class type in call not subclass of class type in signature`
  Est levée si, lorsqu'une méthode demande une instance de classe en paramètre, la classe appelée n'est pas une sous-classe de la classe donnée en signature.

3.4 Erreurs à l'éxecution