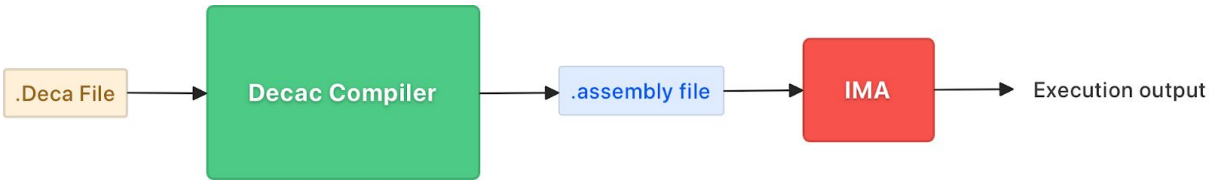
Manuel utilisateur Compilateur **DECAC**

Projet Génie Logiciel 2020

Groupe n° 13

MORIN Lucas
NAVARRO Jérémy
ODEH Majd
OUDOUMANESSAH Geoffroy
POUGET Sylvain

Vue d'ensemble



DECAC est un compilateur destiné à concevoir du code assembleur pour machine IMA à partir de langage DECA. Pour compiler le compilateur **DECAC** il suffit de taper dans le terminal (à la racine du projet):

```
$ mvn compile
```

Pour effacer les exécutables et les fichiers générés après la compilation il suffit de taper dans le terminal (à la racine du projet) :

```
$ mvn clean
```

On supposera dans la suite de ce document, que **DECAC** est à jour, compilé et installé sur la machine cible ainsi que la cible **DECAC** et correctement configuré.

DECAC compile des fichiers `<nomFichier>.deca` en fichier assembleur `<nomFichier>.ass`. Cette génération se fait en ligne de commande:

```
$ deca file_name.deca
```

Après l'exécution le fichier assembleur se fait en ligne de commande:

```
$ ima file_name.ass
```

Les options de DECAC:

```
Decac [ [-p | -v] [-n] [-r X] [-d]* [-P] [-w] [-mt] <fichier deca>... ] | [-b]
```

- ➢ La commande `decac` seul permet d'afficher l'aide.
- ➢ `decac` peut être utilisé sur un ou plusieurs fichiers `.deca`.

Les options possibles sont:

- ❖ **-b** (banner) : affiche une bannière indiquant le nom de l'équipe.
- ❖ **-mt** (methods table) : affiche la table des méthodes.
- ❖ **-p** (parse) : arrête decac après l'étape de construction de l'arbre, et affiche la décompilation de ce dernier. Pour le cas d'un seul fichier source à compiler, la sortie est un programme deca syntaxiquement correct.
- ❖ **-v** (vérification) : arrête decac après l'étape de vérifications, aucune sortie est produite en l'absence d'erreur.
- ❖ **-n** (no check) : supprime les tests de débordement à l'exécution. (débordement arithmétique / mémoire et déréférencement de null)
- ❖ **-r X** (registers) : limite les registres banalisés disponibles entre R[[0, X-1]], avec 4 <= X <= 16.
- ❖ **-d** (debug) : active les traces de debug. Répéter l'option plusieurs fois pour avoir plus de traces.
- ❖ **-P** (parallel) : s'il y a plusieurs fichiers sources, lance la compilation des fichiers en parallèle (pour accélérer la compilation).

Les options '-p' et '-v' sont incompatibles.

Limitations du compilateur DECAC

DECAC est un logiciel qui a pour vocation d'évoluer dans le temps. Ce projet open-source hébergé sur ensimag.gitlab a été construit pour être amélioré dans le temps. Ainsi, il possède à l'heure actuelle quelques limitations.

Gestion des casts d'objet

Le langage deca implémente l'utilisation de cast entre objet hérités. Cependant, DECAC gère le cast d'objet en modifiant directement l'objet cible. Il existe donc des cas dans lesquels le comportement produit n'est pas celui attendu.

```
class A {}
class B extends A {
	void method(){}
}
{
	B b = new B();
	A a = (A)(b);

	b.method();
}
```

Dans ce cas, l'objet b devient un élément de la classe A.
Il est donc impossible d'appeler la méthode method sur le pointeur b, cela produit l'erreur:

```
** IMA ** ERREUR ** Ligne 63 :
BSR avec operande : adresse memoire
```

Comparaison avec null

DECAC n'implémente pas la comparaison avec `null`.

```
{
	Object a = null;
	if (a == null){
		println("true");
	} else {
		println("false");
	}
}
```

Produira une erreur:

```
<fileName>:3:8: Contextual error : Operands for a Comparison should be
the same type
```

En fait, null n'est pas considéré comme une classe avec DECAC. Il ne peut donc pas comparer deux éléments de types différents.

Cast sur des booléens

```
{
      boolean x = (boolean)(true);
}
```

```
fr.ensimag.deca.tools.DecacInternalError: Identifier boolean is not a
class identifier, you can't call getClassDefinition on it
```

Un type booléen n'est pas considéré comme type pour un cast. Ainsi DECAC interprète cela comme un appel sur la méthode boolean.

Conversion INT vers FLOAT lors d'appel de méthode

Un appel de méthode attendant un flottant comme paramètre, appelé avec un entier va générer une erreur.

```
class A {
      void affiche(float f) {
            println(f);
      }
}
{
      A a = new A();
      a.affiche(5);
}
```

Génère:

```
<fileName>:3:8: Contextual error, the parameters given don't respect the
method signature
```

Lors de l'appel de la méthode, chaque arguments doit avoir un type égal à celui de la signature. Or, INT et FLOAT ne sont pas de même type, le compilateur génère alors une erreur contextuelle.

Les interactions avec le compilateur

Comment comprendre les exceptions levées par le compilateur ?
Premièrement, il faut savoir que les exceptions générées par le compilateur sont de la forme suivante :

```
<nom de fichier>.deca:<ligne>:<colonne>: "Message d'erreur"
```

---

1 - Les messages d'erreurs lexicales

Ces erreurs sont levées par l'analyse lexicale du programme.
Si une erreur de ce type est levée c'est que le programme à compiler utilise des caractères non reconnus par le compilateur.

Liste des messages possibles :

- token recognition error at: ' '
  Lors de l'utilisation d'un symbole non reconnu.

---

2 - Les messages d'erreurs de syntaxe

Ces erreurs sont levées par l'analyse syntaxique du programme.
Si une de ces erreurs est levée c'est que le programme à compiler utilise un enchaînement de symboles non reconnu par le compilateur.

Liste des messages possibles :

- Include file not found
  Si un fichier à inclure n'est pas trouvé.

- left-hand side of assignment is not an lvalue
  Si la partie gauche d'une affectation n'est pas une expression "simple". Par exemple, on essaye de faire 3 + x = 5.

- missing ' ' at ' '
  Si il manque quelque chose pour que la séquence de symboles soit valide. Par exemple, on utilise le mot clé "class" seul, sans indiquer de nom de classe.

- mismatched input ' ' expecting { }
  Si un symbole est proposé mais ne correspond pas à ce qui était attendu. Par exemple, on utilise des mots clés "else", "else if" ou "while" seuls.

- no viable alternative at input ' '

- extraneous input ' ' expecting { }

2-1 - Autres erreurs

- Circular include for file <fileName>
  Lors de l'utilisation récursive du #include.

- syntax error, this value is too large
  Lors de l'utilisation d'entiers ou de flottants trop grands (supérieurs à $2^{31} - 1$).

---

3 - Les erreurs de contexte

Ces erreurs sont levées lors de l'analyse contextuelle du programme. Ces erreurs identiques un non respect de la syntaxe en contexte du langage Deca. Pour repérer ces erreurs, le compilateur réalise trois passes.

Liste des messages d'erreurs possibles :

- Contextual error : identifier not defined
  Lors l'utilisation d'un identifiant (variable, classe, champ, méthode) non défini.
  Lors de l'affectation d'une variable sans nom.

- Contextual error : Types don't match!
  Lors de la mauvaise utilisation de type statique. Par exemple, lors du stockage d'un résultat d'une division dans un entier.

- Contextual error : It is not allowed to access a protected field outside a class
  Lors de l'appel d'un champ protected dans une instance main.

- Contextual error with an expression of type 'instance.field' 3.66
  Lors d'un accès abusif à un champ protected dans une classe.

- Contextual error : Type not defined
  Lors de l'utilisation d'un type non défini.
  Lors de l'utilisation d'une classe non défini.

- Override method with different signature
  Lors de la redéfinition d'une méthode en utilisant une signature différente. Les types ou le nombre des paramètres n'est pas celui défini dans la classe mère.

- Unary Minus Operand should be boolean
  Lors de l'utilisation d'un signe "-" sur des opérandes non entières ou flottantes.

- Contextual error, instanceof must be called with an instance and a class
  Lors de l'appel de l'instruction instanceof avec des types primitifs.

- Contextual error, the method called expected <nbParamsExpected> arguments but <nbParamsGiven> were given
  Lors de l'appel d'une méthode sans donner le bon nombre de paramètre attendu.

- Contextual error : Class name was already defined
  Lors de la redéfinition d'une classe.

- Contextual error : Field already defined in this class
  Lors de la redéfinition d'un champ dans une classe.

- Contextual error : Method already defined in the class

  Lors de la redéfinition d'une méthode dans une classe.

- Contextual error : variable already defined

  Lors de la redéfinition d'une variable dans un bloc.

- Contextual error : Return value is expected is this method

  Lorsqu'une méthode autre que de type void, ne possède pas de return.

- Contextual error : Return type can't be "void" (3.24)

  Lors de l'utilisation d'un return dans une méthode de type void.

- Contextual error : Field can't be void

  Lors de la définition d'un champ de type void.

- Contextual error : Variable can't be void

  Lors de la définition d'une variable de type void.

- Contextual error : Parameter can't be void

  Lors de la définition d'un paramètre de type void.

- Contextual error : Condition should be a boolean

  Lors de l'utilisation d'une condition qui n'est pas de type booléenne.

- Contextual error, the parameters given don't respect the method signature

  Lors de l'appel d'une fonction avec des paramètres de types différents que ceux attendu dans sa définition.

- Contextual error : 'this' must be used inside a class

  Lors de l'utilisation de "this" dans le main.

- Contextual error : expecting a class type on the left side
- Contextual error : Incompatible class types

  Lors d'une interaction entre des instances dont les types ne sont pas compatibles

- Contextual error : Operands for a Boolean operation should be booleans

  Lors d'une opération booléenne entre deux opérandes non booléennes.

- Contextual error : Operands for an Arithmetic operation should be int or float

- "Contextual error : Operands for a Comparison should have the same type"

- Contextual error : Operands for an Inequality must be numeric numbers"
  Lors d'une comparaison autre que l'égalité ou la différence, il faut que les opérandes soient des entiers ou des flottants.

- Contextual error : only integers, floats and strings can be printed
  Lors d'un appel à la méthode print, il faut donner en paramètre un ou des flottants, entiers ou chaîne de caractères.

- Contextual error : super is not a class
  Lors de la création d'une classe dont la classe mère déclarée n'est pas une classe.

- Contextual error : Field already defined in this class

- Contextual error : Field can't be void

- Contextual error : Method already defined in the class

- Contextual error : Return type should be the same or subtype of the super class method
  Lors de la redéfinition d'une méthode, il faut vérifier que le type de retour de la nouvelle méthode soit bien celui de la méthode à redéfinir.

- Contextual error : Operands for Divide operation should be int or float

- <identifier> <name> is not a < > identifier, you can't call get< > Definition on it

- Contextual error, it is not possible to call a method on a object that is not a class instance

- Contextual error : Operands must be int [Modulo] int

- Contextual error : The identifier is not a class
  Lors d'un appel à New non suivi d'une classe.

- Contextual error : Unary Minus Operand should be int or float
- Contextual error : Current return type doesn't match original function type

- Contextual error : The left member of 'instance.field' must be the instance of a class

4 - Les erreurs de compilation

- fr.ensimag.deca.tools.DecacInternalError: Identifier affiche is not a field identifier, you can't call getFieldDefinition on it
  Lors de l'appel d'une méthode sans utiliser des parenthèses.

- Stack overflow while compiling file <fileName>
  Lors d'un dépassement de la capacité de la pile.

5 - Les erreurs d'exécution

- Error: heap_overflow
  Lors du dépassement de la capacité du tas (débordement mémoire).

- Error: stack_overflowed
  Lors du dépassement de la capacité de la pile (débordement mémoire).

- Error: Zero_division
  Lors d'une division par 0.

- Error: invalid_input
  Lors de la saisie d'un type différent que celui attendu.

- Error: dereferencement_null
  Lors du déréférencement du pointeur null.

- Error: Float_overflow
  Lors d'un débordement arithmétique

6 - Les erreurs ima

D’autres erreurs peuvent être levées par ima.

Extension Trigo :

Méthode d’utilisation :

Pour utiliser les méthodes de l’extension, on fait appel à la bibliothèque **Math.decah** en utilisant `#include "Math.decah"` :

```
#include "Math.decah"
 {
   Math m = new Math();
   float a = m.sin(0.5);     // 4.79426e-01
   float b = m.cos(0.5);     // 8.77583e-01
   float c = m.atan(0.5);     // 4.63648e-01
   float d = m.asin(0.5);     // 5.23599e-01
   float e = m.ulp(1.0);     // 5.96046e-08
 }
```

Vous pouvez utiliser la bibliothèque Math.decah avec toutes les options du compilateur decac.

La classe Math contient les méthodes :

- sin
- cos
- atan
- asin
- ulp

La classe Utility contient les méthodes :

- sqrt
- floor
- bound

- Pour sqrt() un message d'erreur s'affiche en cas de passage d'un argument négatif.
- De même, un message s'affiche si l'argument donné pour arcsin n'est pas compris entre -1 et 1.

Ordre de grandeur erreurs de l'extension :

Pour un argument compris entre -pi et pi, on estime que l'erreur se situe dans la dernière décimale affichée liée à un arrondi des autres décimales.

Pour des grands nombres, l'erreur est plus importante. Mais ceci est dû à l'ULP très élevé des nombres en question. Pour un argument de l'ordre de 10 000 000 seules les deux première décimale peuvent considérés comme justes.
Au delà de 80 000 000 seule première décimale est correcte. Au delà de 100 000 000 le résultat n'a pas de pertinence.