Manuel utilisateur

Groupe 35

Janvier 2017

1 Fonctionnement du compilateur Decac

Le programme "decac" est un compilateur Deca complet. On peut utiliser un, ou plusieurs fichiers d'entrée par des chemins de la forme : <répertoires/nom.deca>.
Le résultat se trouvera, sauf erreurs, dans un ou des fichiers : <répertoires/nom.ass>.

Le compilateur peut s'utiliser avec différentes options:

- -b (banner) : affiche une bannière indiquant le nom de l'équipe.
- -p (parse) : arrête decac après l'étape de construction l'arbre, et affiche la décompilation de ce dernier (i.e. s'il n'y a qu'un fichier source à compiler, la sortie doit être un programme deca syntaxiquement correct).
- -v (verification) : arrête decac après l'étape de vérifications (ne produit aucune sortie en l'absence d'erreur).
- -n (no check) supprime les tests de débordement à l'exécution:
  - débordement arithmétique
  - débordement mémoire
  - déréférencement de null
- -r X (registers) : limite les registres banalisés disponibles à R0 ... RX-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)

Vous pouvez lancer la commande "decac" sans arguments pour afficher une liste détaillée de ces différentes options.

2 Les messages d'erreur

Formatage des messages d'erreur:

Tous les messages d'erreur que vous pourriez rencontrer lors de l'utilisation du compilateur sont formatés de la manière suivante :

<nom de fichier.deca>:<ligne>:<colonne>: <description informelle du problème>

Dans cette partie, nous allons détailler les différentes erreurs relevées par le compilateur:

2.1 Erreurs lexicales

- "token recognition error at:<caractère(s) problématique(s)>".
  Erreurs rencontrées lorsqu'un caractère ou une suite de caractères n'appartiennent pas au vocabulaire du langage Deca.

2.2 Erreurs syntaxiques

- "mismatched input <caractère problématique> expecting {<Liste des caractères possibles attendus à la place>}"
- "extraneous input <caractère problématique> expecting {<Liste des caractères possibles attendus à la place>}"
- "missing <caractère attendu> at <caractère problématique>"
- "no viable alternative at input <caractère problématique>"

  Erreurs rencontrées lorsqu'un programme ne respecte pas la syntaxe du langage Deca.
- "Overflow error"

  Erreurs liées au débordement arithmétique d'entiers ou de flottants. Cela peut se produire si le programme contient un nombre trop grand ou ,au contraire, trop proche de 0.

2.3 Erreurs contextuelles

- "Identifier has no attached Definition"
  Si vous utilisez une variable non déclarée, ou déclarée après.
- "The field is protected"
  Si vous tentez d'accéder à un attribut protégé.
- "Invalid assignement"
  Si vous essayer d'attribuer à une variable une expression d'un type qui ne peut pas correspondre au type de la variable.

- "Field expected"
  Si vous oubliez les parenthèses en appelant une méthode.
- "Return type not compatible"
  Si vous retournez un type qui ne correspond pas au type de retour défini lors de la définition de la méthode.
- "Impossible cast"
  Le cast que vous essayez de faire entre deux types n'est pas réalisable.
- "Types not compatible for comparison"
  Vous ne pouvez pas faire de comparaisons avec ces deux types.
- "Types not compatible for exact comparison"
  Vous ne pouvez pas faire de comparaisons exactes avec ces deux types.
- "Variable already defined in the current environment"
  Vous essayez d'initialiser une variable déjà initialisée dans l'environnement courant.
- "Type undeclared"
  Vous essayez d'utiliser un type qui n'existe pas.
- "Invalid type for unary minus"
  Vous essayez d'appliquer un moins unaire à un type non compatible.
- "Modulo applies only on int"
  Le modulo ne s'applique qu'à des entiers.
- "Type not compatible with arithmetic operator"
  Vous utilisez un opérateurs arithmétique sur des types différents de "int" ou "float".
- "The expression is not a condition"
  Vous devez utiliser une expression de type "boolean".
- "Return type is not compatible for override"
  Le type de retour n'est pas le même que celui de la fonction que vous essayez de réécrire depuis une classe fille.
- "Method with different signature already declared"
  Vous ne pouvez pas surcharger une méhode. Autrement dit, vous ne pouvez pas écrire une méthode avec un nom déjà existant et une méthode différente
- "A method with the same name has already been declared in the class"
  Vous essayez de réécrire une méthode avec le même nom et la même signature qu'une méthode déjà existante.

- "Name of the method already used for an other thing"
  Le nom de méthode que vous essayez d'utiliser a déjà été utilisé pour initialiser une variable ou une classe.
- "Name of the method used is a type"
  Le nom de méthode que vous essayez d'initialiser est un type.
- "Incompatible type for print"
  Vous essayez d'utiliser print sur un type différent de "int","float" et "boolean".
- "this call impossible in main"
  Vous ne pouvez pas utiliser "this" à l'intérieur du main.
- "Impossible InstanceOf, type of expression is not compatible"
  L'expression que vous testez avec InstanceOf doit être soit nulle soit une classe.
- "Impossible InstanceOf, type is not a class"
  Le type que vous rentrez dans InstanceOf doit être une classe.
- "Class already defined"
  Vous essayez de définir une classe qui a le même nom qu'une classe déjà existante.
- "Super class is not defined"
  La classe que vous essayez d'hériter est nulle.
- "SuperClass name is not type of class"
  Le nom de la classe mère n'est pas une classe
- "Field name used is a type"
  Vous essayez d'initialiser un attribut avec le nom d'un type.
- "Field already defined"
  Le nom d'attribut que vous essayez d'utiliser a déjà été attribué à un autre attribut.
- "Parameter name used is a type"
  Vous essayez d'initialiser un paramètre avec le nom d'un type.
- "Parameter with same name already defined"
  Le nom de paramètre que vous essayez d'utiliser a déjà été attribué à un autre paramètre.
- "Not enough arguments for this method"
  La méthode que vous appelez nécessite plus d'arguments.
- "Too many arguments for this method"
  La méthode que vous appelez nécessite moins d'arguments.
- "Class not found"
  La classe que vous essayez d'initialiser n'existe pas.

2.4 Erreurs d'éxécution

Certaines erreurs peuvent être levées au moment de l'éxécution du code assembleurs:

- "function without return" Cette erreur est levée si une fonction avec un type de retour différent de void se termine sans avoir exécuté de "return".
- "error from class conversion" Si vous essayer de faire un Cast d'une classe mére en sa classe fille mais que les espaces mémoires des deux objets ne correspondent pas.
- "null dereferencing" Si vous tentez d'accéder à des méthodes ou attributs d'une variable de type "null".
- "divided by zero" Division par 0.

3 L'extension Math

La librairie Math se trouve dans : src/main/resources/include/library On y trouve l'implémentation de la classe Grand FLoat ainsi que les principales fonctions optimisées de la classe math :

- calcul de π
- calcul de $\pi/2$
- sinus
- cosinus
- tangeante
- arctan
- arcsin
- arccos
- ulp

ainsi que les fonctions nécessaires pour leur implémentation.

src/main/resources/include/deca contient d'autres méthodes de calcul de ces fonctions que nous avons pas choisi au final.

src/main/resources/include/java contient la version Java.

3.1 Limites de l'extension

Nous détaillons ici la précision atteinte par nos fonctions exprimée en ulp. Ces précision sont celles relevées le Lundi 23/01 à 13h. Nous développons encore quelques idées qui peuvent les améliorer davantage. Concernant les fonctions basiques, on a implémenté les fonctions max, min, puissance, factoriel et signe avec une précision parfaite. Pour les fonctions trigonométriques nous avons travaillé de façon agile, en implémentant les algorithmes qui nous paraissaient le plus logique, en effectuant un travail d'auto-critique puis en développant une nouvelle version. Au final nous avons obtenu des erreurs de l'ordre de l'ulp en Java mais en passant en Deca nous perdons de la précision. Des hypothèses sur l'explication de cette perte sont analysées dans le rapport final de l'extension. Voici une description des erreurs pour les fonctions implémentées (on a pris en compte la version la plus précise pour chaque fonction):

3.1.1 Sinus

On a réalisé des tests dans l'intervalle $[-\pi, \pi]$: pour la version Java on obtient des erreurs entre 0 et 1 ulp. En Deca l'erreur varie entre 0 et 1000 ulp. Elle atteint son maximum au voisinage de pi. Pour les valeurs qui ne sont pas dans cet intervalle, on a implémenté une fonction qui permet d'adapter ces valeurs et calculer leurs équivalent dans $[-\pi, \pi]$ (décalages de $2 * \pi$) . Cette fonction provoque des débordements dans la pile Déca pour des nombres avoisinants 1000. Nous pensons pouvoir régler cette erreur avant le rendu final de l'extension. Aussi cette fonction effectue forcément aussi des erreurs à chaque soustraction minimisées par les Grand Float.

3.1.2 Cosinus

On a réalisé des tests dans l'intervalle $[-\pi, \pi]$: pour la version Java on obtient des erreurs entre 0 et 2 ulp. En Deca l'erreur varie entre 0 et 3000 ulp. Elle atteint son maximum au voisinage de $\pi/2$. Pour les valeurs qui ne sont pas dans cet intervalle, on fait de même que pour la fonction sinus vu la périodicité de ces fonctions.

3.1.3 Arctan

Les tests pour cette fonctions ont été réalisés dans l'intervalle $[-10.10]$: pour la version Java on obtient des erreurs entre 0 et 3 ulp mais en Deca l'erreur varie entre 0 et 90 ulp. Cette fonction est donc précise.

3.1.4 Arcsin

On a réalisé des tests dans l'intervalle $[-1, 1]$ qui est son domaine de définition : pour la version Java on obtient des erreurs entre 0 et 9 ulp mais en Deca l'erreur augmente beaucoup et varie entre 0 et 800 000 ulp mais les valeurs restent bien

cohérentes. On compte améliorer cette fonction et trouver un algorithme qui est plus précis en Deca et cette fois on va tester directement sur Deca.

3.1.5 Autres fonctions

Pour le reste des fonctions trigonométriques , on a implémenté la fonction tangente et arcsinus avec des formules simples mais ces fonctions sont moins précises vu qu'elles utilisent déjà les autres fonctions: $tan(x) = sin(x)/cos(x)$ et $arccos(x) = \pi/2 - arcsin(x)$.