Manuel Utilisateur

Projet GL Equipe 16
Amine ANGAR
Dy EL ALEM
Mouad TARBOUI
Oussama KADDAMI
Youness LEHDILI

1 Description du compilateur

Un compilateur pour le sous-langage Deca de JAVA qui produit des fichiers assembleurs ".ass" exécutables avec la machine IMA.

1.1 Fonctionnalités implémentées

Ce compilateur implémente toutes les spécifications du langage Deca, et la bibliothèque mathématique de l'extension TRIGO.

1.2 Guide d'utilisation

Les fichier source ".deca" sont compilés à l'aide de la commande decac, la syntaxe de l'utilisation de la commande decac est la suivante :

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

La commande dacac sans options produit simplement le fichier assembleur ".ass".

```
decac <fichier deca>
```

Ce fichier est exécutable par la machine IMA via la commande ima.

```
ima <fichier ass>
```

Options du compilateur:

- -b (banner) : affiche une bannière indiquant le nom de l'équipe ;
- -p (parse) : arrête decac après l'étape de construction de 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);
- -r X (registers) : limite les registres banalisés disponibles à $R_0 ... R_{X-1}$, avec $4 \leq X \leq 16$;
- -P (parallel) : s'il y a plusieurs fichiers sources, lance la compilation des fichiers en parallèle (pour accélérer la compilation);

2 Limitations:

La partie génération de code et l'ensemble du compilateur permettent d'implémenter toutes les fonctionnalités sauf l'inclusion d'un code assembleur. Nous avons prévu de le faire le dernier jour, mais les différents problèmes révélés lors de l'exécution de la partie extension trigo par notre compilateur ne nous ont pas permis de le faire. Nous avons essayé de détecter et de corriger le plus grand nombre d'erreurs révélées. Il reste des problèmes lors de certaines opérations qui stipulent un cast implicite entre flottants et entier. Nous n'avons découvert certains de ces défauts qu'aux derniers moments et leur correction risquait de nuire aux autres parties. Les options -n et -d du compilateur n'ont pas aussi été implémentés.

3 Messages d'erreurs

3.1 Erreurs lexicales et syntaxiques:

Les entiers que deca accepte sont ceux qui sont représentables sur 32 bits. Quant aux littéraux flottants, le compilateur refuse les flottants infinis (trop grand) levant une exception "InfiniteFloatException" , les flottants dont l'arrondi se fait vers zéro "NullRoundingException" et les flottants qui ne sont pas des nombres "NaNFloatException". Le mot clé public ne peut pas être déclaré en deca . La visibilité "protected" est par contre acceptée.
D'autres erreurs sont traités par Antlr:

- missing '{' at 'extends' : Deux extends dans une déclaration de classe ;
- token recognition error at : Chaine de carcactères icomplète;
- no viable alternative at input : Pas de type de retour de la méthode;
- Infinite float error : Un float infini;
- mismatched input : instanceof incorrecte;
- mismatched input : Instruction en dehors du main

3.2 Erreurs Contextuelles

Les erreurs de syntaxe contextuelles sont signalées par l'exception ContextualError qui s'affichent sous la forme nomFichier:numeroLigne:nueroColonne: message d'erreur. Voici la liste des messages d'erreur et les configurations qui les provoquent:

Erreurs liées aux déclarations

- undefined identifier "nom" : on utilise une variable qui n'a pas été déclarée;
- Variable "name" is already declared : Une variable du même nom a déjà été déclarée;
- The name "name" is already used : Un attribut ou une méthode du même nom a été déclaré dans la classe courante;
- class already defined : Une classe du même nom est déjà déclarée;
- The class "class name" is not defined : la classe utilisée n'est pas définie;
- incompatible types for assignement : initialization incorrecte;

Erreurs liées aux opérandes

- Arithmetic operation not defined for the used types : opération arithmétique sur des type incompatibles;
- Incompatible types for binary boolean operation : opérateur boolean non défini pour les type utilisés;
- The binary operation used is not defined for the operands types : opération de comparaison non définie pour les types utilisés;
- Not operator incompatible whith this expression : l'utilisation de Not sur un type incompatible;
- Unary operator - incompatible whith this expression : l'utilisation du - unaire sur un type incompatible;

Erreurs liées aux sélections et appels de méthodes

- undefined identifier "nom" : on utilise un attribut qui n'a pas été déclaré;
- inaccessible protected field : le champs protégé n'est pas accessible;
- undefined method identifier : on utilise une méthode qui n'a pas été déclarée;
- the number of parameters is incorrect : l'appel d'une méthode avec un nombre de paramètres incorrect;

- The declaration of "method name" is incompatible with its declaration in superclass : redéfinition incorrecte de la méthode;

Autres erreurs

- Cast impossible : cast incorrect;
- incompatible types with instanceof operation : les types utilisés pour instanceof ne sont pas compatibles avec cet opérateur;

3.3 Erreurs à l'éxécution du code assembleur

Le compilateur implémenté permet les erreurs liées à la compilation et l'exécution. Nous avons essayé d'implémenter un compilateur qui permet de détecter les erreurs selon la spécification. Concernant la partie C, les erreurs qui peuvent avoir lieu sont :

- Stack overflow: L'instruction TSTO est positionné au début de chaque bloc et permet de détecter un cas qui cause le débordement de la pile. Un exemple de test qui permet d'illustrer cela est présent dans les tests invalides. Message de sortie associé: "Error: Stack Overflow";
- Heap overflow: Le débordement du tas suite à l'instanciation d'objets dépassant la capacité de la pile est normalement géré par le compilateur. Message de sortie associé : "Erreur : heap_over_flow";
- IO exception: Les erreurs liées aux fonctions d'entrées-sorties sont également détectées par le compilateur. En particulier, ceci pourra être testé dans la partie interactive. Cela concerne principalement le cas ou le type de la valeur fournie par l'utilisateur n'est pas conforme au type attendu(float au lieu de int ou l'inverse). Message de sortie associé: "Error: Input/Output error";
- Débordement liée aux opérations arithmétiques sur les flottants: Le débordement lié aux opérations sur les flottants est testé sur toutes les opérations arithmétiques. La division par rapport à 0 est donc un cas particulier de ce type de débordement. Message de sortie associé: "Error: Débordement arithmétique sur flottants";
- Division entière sur 0: À l'encontre des opérations sur flottants, les débordements sur des opérandes entiers n'engendre des erreurs que dans le cas d'une division part 0. Pour ce faire, notre algorithme compare le dénominateur d'une division entière à 0, et lève une exception dans le cas d'égalité. Message de sortie associé: "Error: Division entière par 0";
- Référencement nul: Dans le cas d'une variable de type classe ne contenant pas une adresse d'un objet instancié(dans le tas), le compilateur lève une exception. Cela correspond donc pour la partie génération du code à une

référence la valeur “null”, puisque nous affectons par défaut cette valeur aux variables de type classe(Objet) non initialisée. Message de sortie associé: ”Erreur : dereferencement de null”;

- Cast exception: Les erreurs liées à un cast non conforme au type sont détectées par le compilateur. Message de sortie associé: ”Erreur; le cast n'est pas possible”

4 Utilisation de l'extension

Les fonctions implémentées dans le fichier Math.Decah sont :

1. _min(float a, float b): renvoie le minimum entre les flottants passés en argument;
2. _max(float a, float b): renvoie le maximum entre les flottants passés en argument.
3. fact(int n): renvoie le factoriel de l'entier passé en argument;
4. pow(float a, int b) : lève a à la puissance de b et renvoie le résultat sous la forme flottant;
5. _Ulp(float x) : renvoie la taille d'un Ulp de l'argument;
6. _fmod(float x, float y) : calcule le reste de la division entre deux flottants;
7. _abs(float a):renvoie la valeur absolue d'un flottant;
8. _signe(float z) : renvoie le signe d'un flottant;
9. _cos(float x): calcule et renvoie le cosinus du flottant passé en argument;
10. _sin(float x) :calcule et renvoie le sinus du flottant passé en argument;
11. _asin(float x):calcule et renvoie le arcsinus du flottant passé en argument;
12. _sqrt(float a):calcule et renvoie le racine carrée du flottant passé en argument;
13. _atan(float x) :calcule et renvoie le arcstangente du flottant passé en argument.

5 Fonctionnement de l'extension

l'ajout de #include ”Math.decah” au début du fichier permet d'utiliser les fonctions indiqués si dessous.
La fonction arcsin doit prendre en argument un flottants compris entre -1 et 1,il faut aussi faire attention aux types des paramètres(float dans la majorité des

cas)
les fonction _cos et _sin donnent des résultats identiques à une dizaine d'ulp prés pour des valeur proches de pi/2
pour des flottants très grands, la fonction _fmod donne des résultats imprécis, ce qui peut influencer le résultat des fonctions _cos et _sin,
Ces méthodes sont des approximations de valeurs mathématiques, dont on mesure la précision grâce aux ULP (Unit in the Last Place).
Pour une analyse complète et détaillée des algorithmes et de leurs précisions respectives, il faut se reporter à la documentation de la classe Math.