Architecture pour l'étape C

Equipe 17

Grenoble, 16 janvier 2017

I. Liste des composants

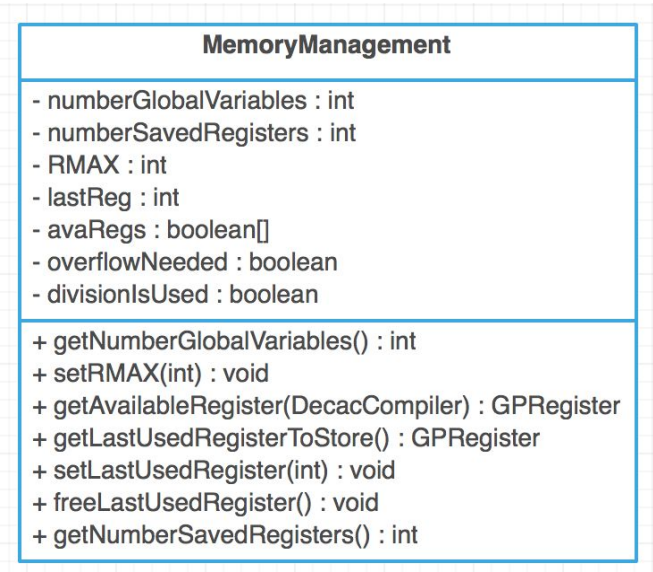
- src/main/java/fr/ensimag/deca/tree/*
- src/main/java/fr/ensimag/deca/codegen/
    - MemoryManagement.java: la gestion des registres et la pile
    - CodeGenInst.java: l'implémentation de code non-triviales
- src/test/java/fr/ensimag/deca/codegen/*
    - ManualTestGencode.java: l'affiche le code d'assembleur qui sera généré
- src/test/deca/codegen/ : Cas de tests en Deca

II. Description

L'objectif de l'étape C est de générer du code assembleur à partir d'un arbre d'un programme Deca contextuellement correct. Les principales problèmes dans ce travail sont la gestion des registres et la pile, la gestion des erreurs d'exécution (division par zéro, l'erreur d'entré/sortie,..)

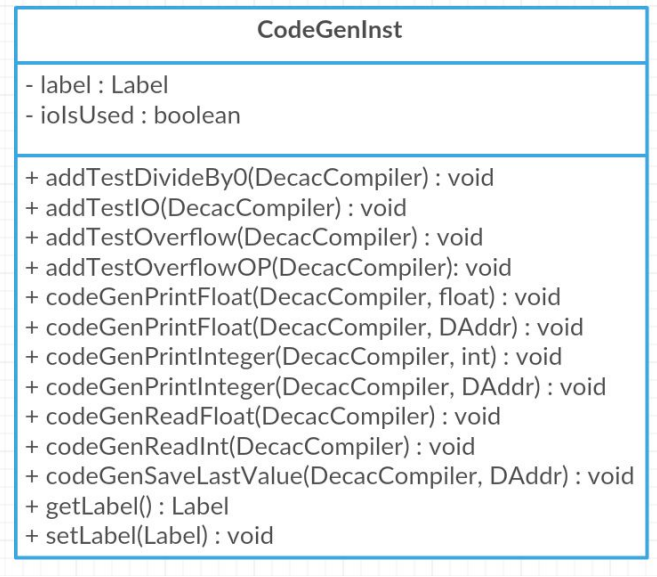
III. Spécification des composants

1. MemoryManagement.java:



Pour cette classe, on gère les registres par un tableau des booléennes *avaRegs* qui montre l'état des registres. Les méthodes concernées sont implémentées pour récupérer les informations nécessaire dans la génération du code

2. CodeGenInst.java:



On ajoute de code d'assembleur non-triviales par les methodes codeGen*. Les méthodes addTest vont ajouter les vérifications pour générer des erreurs si nécessaire (par exemple, si on n'a pas des entrées, c'est pas nécessaire d'ajouter les vérifications).

3. Gestion des erreurs

On ajoute de code de gestion des erreurs après la génération du code de Main. En faisant des tests concernés, on lève des erreurs d'exécution avec un message

Erreur	Message	Test concerné
stack_overflow_error	Error: Stack Overflow	Test de débordement de pile
overflow_error	Error: Overflow during arithmetic operation	Test de débordement en faisant la multiplication deux floatants
division_by_zero_error	Error: division by zero	Test de division par 0 en faisant la division ou le modulo par 0
io_error	Error: Input/Output error	Test d'entré/sortie en lisant un floatant ou un entier depuis console

4. codeGenInst:

L'implémentation des méthodes va ajouter les instructions assembleurs directement dans le program. Les instructions sont souvent l'enregistrement la valeur de feuille de l'arbre dans un registre ou l'implémentation d'une opération (add, multiply, devide...