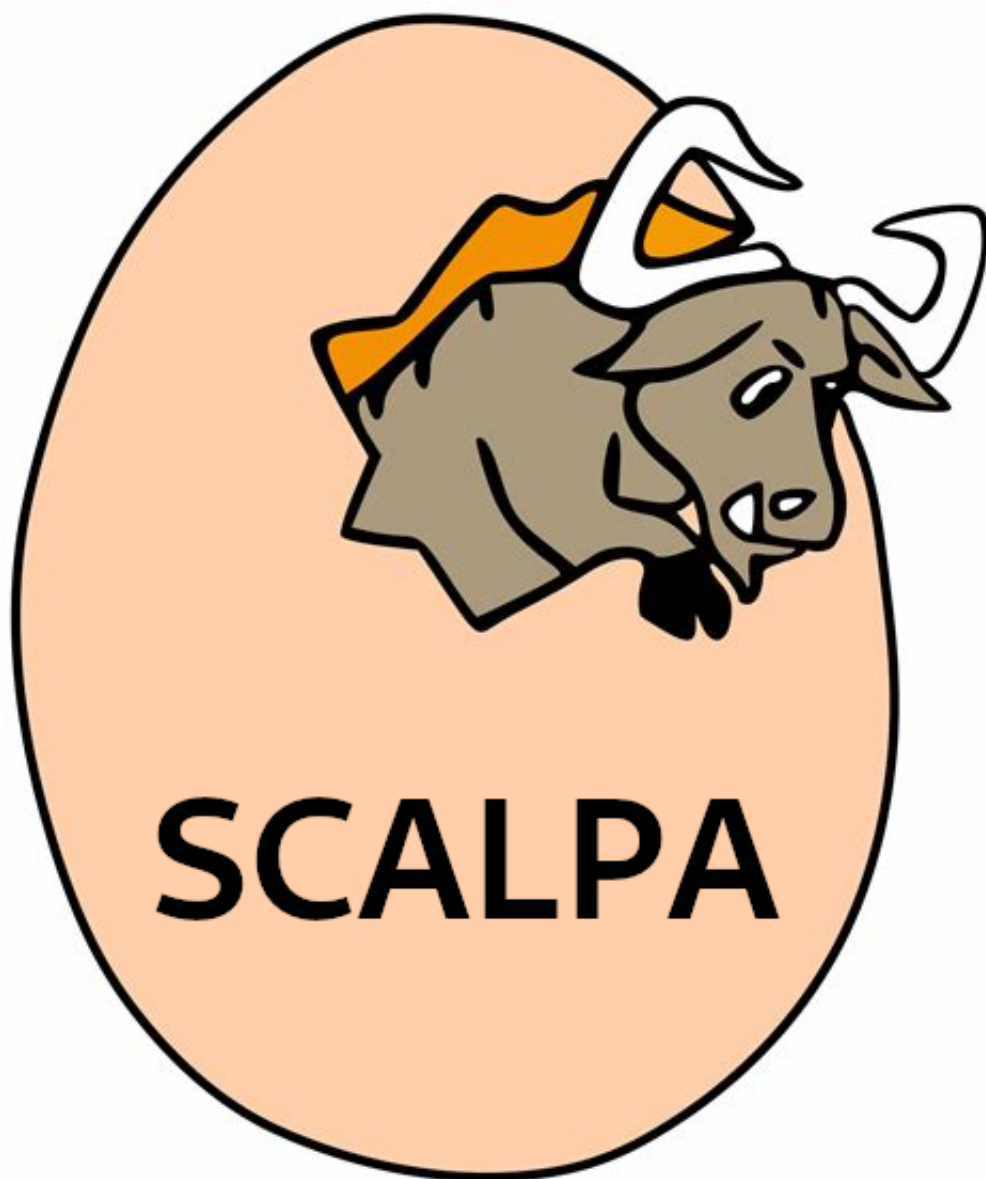


# Rapport du projet de compilateur SCALPA

Danyl El-Kabir  
Jérémy Bach  
Nadjib Belaribi  
François Grabenstaetter

1<sup>er</sup> janvier 2021



## **Table des matières**

1	Introduction	3
2	Capacités de notre compilateur	3

## 1 Introduction

L'objectif de ce rapport est de résumer les capacités de notre compilateur SCALPA. Le compilateur génère du code MIPS à partir d'un code SCALPA. Nous avons réalisé ce projet de manière incrémentale, en implémentant les différentes fonctions au fur et à mesure.

Le projet a donc été séparé en sous tâches : - Analyse lexicale - Implémentation des affectations et opérations arithmétiques - Implémentation des conditions - Implémentation des boucles - Implémentation des tableaux - Implémentation des fonctions

Chaque sous partie peut être représentée comme un sprint dans une méthode de travail agile. En effet, nous avons commencé à développer le compilateur pour qu'il se limite aux affectations de variables ainsi que sur les opérations sur les entiers. Cette étape cruciale nous a permis de nous familiariser avec les mécanismes de génération de code par exemple. Après avoir implémenté les affectations et opérations arithmétiques, nous avons testé le compilateur via un jeu de test. Dès que la sous partie fut finie, nous avons pu passer à la prochaine phase du projet avec l'implémentation des conditions.

C'est ainsi que s'est déroulé le développement de ce compilateur en enchainant systématiquement les phases d'analyses lexicale, syntaxique, de génération de code et de test pour chaque sous partie.

## 2 Capacités de notre compilateur

Le compilateur est donc capable de générer un programme MIPS R2000 (utilisable sur un simulateur de processeur comme SPIM ou Mars) à partir d'un programme écrit en SCALPA. Le programme écrit en SCALPA peut donc supporter :

- Des affectations sur les entiers, les booleens et les chaînes de caractères
- Des opérations sur les entiers ou des variables entières (additions, multiplications, soustraction, division, puissance) tout en respectant les priorités de calcul.
- Des opérations sur les variables booléennes ou des constantes booléennes (OR, AND, XOR, NOT)
- Les opérations de comparaisons sur les entiers sont également implémentés et peuvent être utilisés comme conditions (<, >, <=, >=, =, <>)

Ces opérations de comparaisons permettent de créer des conditions de type *if cond then instr else instr* ou *while cond do instr*. Les conditions sont implémentées et permettent de réaliser toutes sortes de boucles. Les conditions peuvent être des comparaisons sur des entiers, ou des opérations sur des booleens par exemple. Nous pouvons également imbriquer les conditions pour vérifier plusieurs assertions par exemple et distinguer des cas. Il est possible de simuler une boucle for avec un while, mais il aurait cependant été possible de l'implémenter directement.

En ce qui concerne les tableaux, ils sont entièrement fonctionnels et permettent de faire des affectations ou des opérations sur des éléments du tableau en fonction du type de variable du tableau. Les tableaux peuvent contenir des booleens ou des entiers et sont déclarés avec des intervalles, ces intervalles correspondent aux index du tableau sur les dimensions.

Les tableaux à multidimensions sont également fonctionnels à condition d'avoir le même type dans toutes les cases. Le contenu d'une case du tableau est interprété comme une variable de ce même type et permet donc de formuler des conditions et des opérations sur un contenu du tableau.

De plus l'analyse lexicale de notre compilateur se fait sans sensibilité à la casse, ce qui laisse une "marge d'erreur" au programmeur, s'il se trompe et écrit While, wHile ou whiLE...etc ce n'est pas grave notre compilateur saura comprendre son code.

Pour finir, les fonctions peuvent être définies et utilisées dans le programme tout en prenant des arguments en entrée et en retournant des valeurs. Les arguments des fonctions peuvent être des entiers, des booléens ou des tableaux. Les tableaux peuvent être passés en copie ou en référence. Si le tableau est passé en argument par référence, la fonction modifiera le tableau par effet de bord sur le tableau initial. Les fonctions récursives sont également supportées et complètement fonctionnelles.