Matthieu VINCENT

Loïc REISDOERFER

Mervine LIEFFROY

Martin LEMAITRE

**Projet de compilation**

**Rapport**

L3 MIAGE année 2016 / 2017

**Sommaire**

[Introduction 3](#_Toc480712450)

[Définition de notre langage 4](#_Toc480712451)

[Programme qui calcul la somme des n premiers entiers 4](#_Toc480712452)

[Plus petit programme accepté 4](#_Toc480712453)

[Les variables 4](#_Toc480712454)

[Les commentaires 5](#_Toc480712455)

[Lecture et écriture 5](#_Toc480712456)

[La structure de contrôle 5](#_Toc480712457)

[La conditionnelle 5](#_Toc480712458)

[Déclaration et appel de fonction 6](#_Toc480712459)

[Réalisation du projet 7](#_Toc480712460)

[Organisation 7](#_Toc480712461)

[La grammaire 7](#_Toc480712462)

[La table des symboles (TDS) 7](#_Toc480712463)

[L’arbre syntaxique (AST) 7](#_Toc480712464)

[Le générateur UASM 7](#_Toc480712465)

[Les difficultés rencontrées 7](#_Toc480712466)

[Conclusion 8](#_Toc480712467)

# Introduction

Le but de ce projet était de réaliser un compilateur pour un langage de programmation que nous avons nous même créé. Le compilateur doit dans un premier temps vérifier la syntaxe du programme source (écrit dans notre langage) puis générer un programme cible en langage assembleur beta.

Ce principe est le même pour tous les langages de programmation existant. Le schéma suivant décrit simplement le processus de compilation.

Compilateur

Programme cible

Programme source

Compilateur qui permet la traduction du fichier en assembleur.

Fichier contenant le même code mais écrit en assembleur beta.

Fichier contenant du code (écrit avec notre langage).

On peut donc vulgariser la fonction d’un compilateur en disant que c’est un traducteur. C’est cette partie que nous avons donc programmé.

# Définition de notre langage

## Programme qui calcul la somme des n premiers entiers

Voici le programme dans notre langage qui permet de calculer la somme des n premiers entiers.

Exemple pour la valeur 3 : 0+1+2+3 qui vaut 6

*fonction entier sommePremierEntier(n){*

* *loc entier iterateur = 0;*
* *loc  entier resultat = 0;*

*tantQue (iterateur <= n) {*

*resultat = resultat + iterateur;*

*iterateur = iterateur + 1;*

*}*

* *retourner resultat;*

*}*

## Plus petit programme accepté

Voici le plus petit programme que pourra accepter notre langage. Ce bout de code ne produira rien comme résultat et on peut remarquer que tous les programme commenceront pas la fonction principale.

*fonction principal(){*

*}*

## Les variables

Cette partie permet de voir comment les variables sont déclarées dans notre langage. Les variables seront toujours du type entier (seul type que notre programme prendra en compte au début). Une distinction sera faite pour les variables globales, accessible dans tout le programme, et les variables locales, accessible uniquement dans le bloc de code où elles sont déclarées.

*glob entier maVariable ;*

*loc entier maVariable ;*

Afin de faire une affectation de valeur à une variable, il suffit d’utiliser le signe **=**.

## Les commentaires

Notre langage contiendra un système de commentaire comme dans tout autre langage de programmation. Il sera possible de faire des commentaires sur une ligne ou multi-lignes.

*/\**

*\* Commentaire multi-lignes*

*\*/*

*// Commentaire sur une seule ligne*

## Lecture et écriture

Notre langage permettra la saisie de valeur au clavier et aussi l’écriture à l’écran. Pour cela il faudra appeler ces deux méthodes qui seront déjà implémentés avec le langage.

*glob entier maVariable = lire() ;*

*ecrire(maVariable) ;*

## La structure de contrôle

Notre langage proposera une structure de contrôle qui est le tant que. Voici le code de celle-ci :

*tantQue (condition){*

*}*

Une condition peut être une opération du type =, <, >, <=, >=.

## La conditionnelle

Le langage contiendra une structure conditionnelle qui est le si. Voici son code :

*si (condition){*

*}*

Comme défini au dessus, une condition est une opération du type =, <, >, <=, >=.

## Déclaration et appel de fonction

Afin de réaliser des programmes plus complets, notre langage pourra contenir des fonctions (comme dans la première partie de ce document). Voici la déclaration d’une fonction :

*fonction entier nom(paramètre1, paramètre2, ...){*

*// Code de la fonction*

*retourner variable ;*

*}*

Pour appeler une fonction, il suffit de mettre le résultat de la fonction dans une variable.

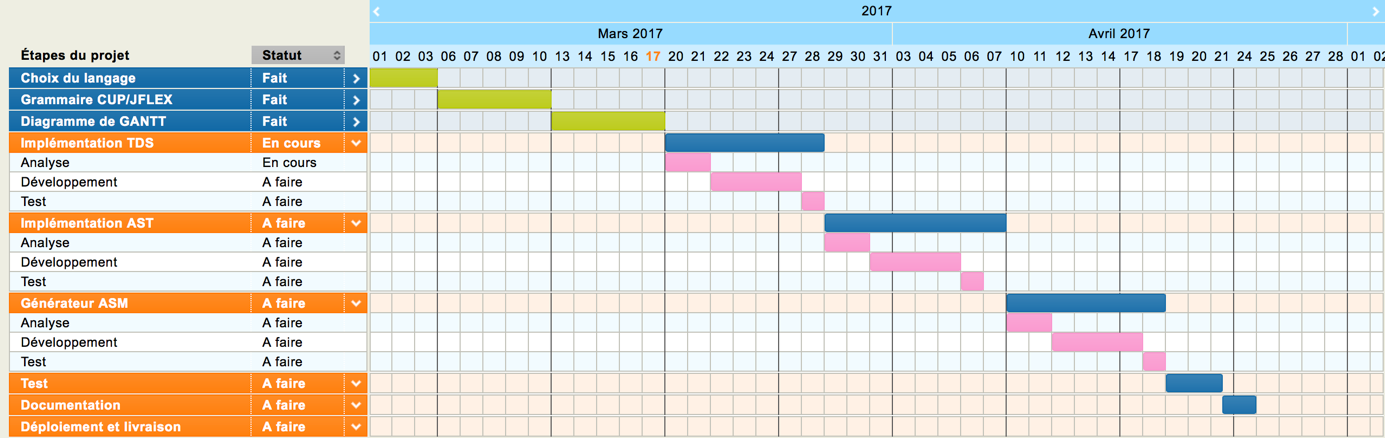
*glob entier maVariable = appel maFonction(paramètre1, paramètre2, ...) ;*

# Réalisation du projet

Dans cette partie nous allons vous expliquer toute la réalisation du projet et les différentes phases de celui-ci. Ensuite nous évoquerons les difficultés que nous avons rencontrées.

## Organisation

Pour bien organiser notre travail, nous avons élaboré un diagramme de GANTT en début de projet afin de spécifier le temps alloué au différentes taches que nous avons identifiées.



Ce diagramme nous a également permis de nous répartir le travail entre nous en fonction des tâches restantes en mettant très régulièrement à jour notre diagramme de GANTT.

## La grammaire

## La table des symboles (TDS)

## L’arbre syntaxique (AST)

## Le générateur UASM

## Les difficultés rencontrées

# Conclusion