Syllabus du module de compilation M1

Jules Chouquet
jules.chouquet@univ-orleans.fr

7 janvier 2025

Table des matières

1 Présentation générale **1**
1.1 Résumé 1
1.2 Répartition des séances de travail 2

2 Références **2**

3 Outils et langages **2**

4 Prérequis **3**
4.1 Langages de programmation 3
4.2 Architecture des ordinateurs 3
4.3 Algorithmique 3
4.4 Conception 4
4.5 Théorie des langages 4
4.6 Systèmes d'exploitation 4

5 Contenu **4**

6 Objectifs **5**

7 Modalités d'évaluation **5**

8 Descriptif prévisionnel des séances de cours magistral **6**

9 Références **6**
9.1 Théorie des langages 6

1 Présentation générale

1.1 Résumé

Ce module présente les différentes composantes d'un compilateur, et les méthodes d'implémentation associées.

Un premier objectif de ce cours est d'avoir une vision d'ensemble de ce qui sous-tend le fonctionnement des programmes, en comprenant comment le code produit dans un langage de haut niveau par le programmeur peut être transformé en quelque chose d'exécutable par une machine. On tâchera de répondre à la question suivante : « que se passe-t-il entre le programmeur et la machine ? »

Le cours n'abordera que la partie *logicielle* de cette transformation : il s'agira de traduire un langage (impératif) en code assembleur. (On ne regardera pas les aspects *matériels*)

Plusieurs domaines de l'informatique seront traités en complémentarité ; la compilation cheminant depuis la théorie des langages jusqu'à l'architecture des ordinateurs.

Ce module sera également l'occasion de développer un programme complexe, avec des méthodes diverses mais complémentaires. L'implémentation d'un compilateur sera notamment l'occasion de renforcer ses compétences en programmation (java), tant sur la conception que sur la manipulation d'objets complexes.

L'évaluation portera notamment sur ce projet, partiellement réalisé en séances de travaux pratiques.

1.2 Répartition des séances de travail

— 14h de cours magistral
— 10h de TP
— 6h de TD

2 Références

L'ouvrage de référence complet *Modern compiler implementation in Java* d'A. Appel (`https://www.cs.princeton.edu/~appel/modern/java/` utilise des concepts communs au cours, notamment liés à un style orienté objet pour la conception du compilateur.

Pour les ouvrages relatifs aux prérequis pour le cours et aux domaines annexes, contactez directement l'enseignant si vous souhaitez un conseil adapté à vos besoins.

3 Outils et langages

Le langage utilisé pour la conception du compilateur est Java (dans ses versions les plus récentes). L'architecture ciblée pour ce compilateur est l'assembleur MIPS32, pour lequel il existe des simulateurs (notamment en ligne de commande et en application web).

Pour l'analyse de langages formels, nous utiliserons notamment ANTLR4 : il faut maîtriser la syntaxe pour la description de grammaires et la génération d'arbre de syntaxe (c'est le même outil qu'utilisé en travaux pratiques du module de langages algébriques. Voir section 4.5).

4 Prérequis

Le sujet du cours est fortement lié à des domaines très variés de l'informatique en général.

Dans cette section sont décrites les connaissances attendues et nécessaires pour aborder le module de compilation. À la fin de chaque sous-section, les identifiants des modules correspondant, enseignés à l'Université d'Orléans sont indiqués. Il sera ainsi possible de se renseigner pour obtenir de l'aide ou des supports de cours dans les enseignements en question. Sera indiquée l'importance de ces modules en tant que prérequis pour le cours de compilation (préférable, important, ou indispensable).

Conseil Si un cours est indiqué en [**essentiel**], assurez-vous d'en maîtriser le contenu de façon solide. S'il est indiqué en [important], il faut connaître les concepts et langages concernés, mais sans forcément être expert du sujet.

4.1 Langages de programmation

Une connaissance des principaux langages de programmation est attendue. En particulier, une maîtrise des paradigmes de langages et de ce qui les distingue est essentielle. Notamment : style impératif, fonctionnel, orienté objet.

Le programme principal étudié et conçu pendant le cours et les travaux pratiques sera basé sur Java et des concepts lié à l'orienté objet. C'est le langage à maîtriser en priorité dans l'écriture de projets structurés.

Par ailleurs, des exemples de cours seront présentés en OCaml et en C, qui sont donc aussi à connaître pour bien assimiler les concepts abordés.

Modules concernés

— Programmation Fonctionnelle (SLA4IF02) [important]
— Programmation impérative et orientée objet (SLA5IF07) [important]
— Programmation orientée objet (SLA3IF03) [**essentiel**]

4.2 Architecture des ordinateurs

Le cours demande une connaissance générale de l'architecture des ordinateurs. En particulier, les questions relatives au langage machine et à l'assembleur seront importantes, et il y aura peu de rappels (et rapides) sur ces thèmes.

Modules concernés

— Architecture des ordinateurs (SLA3IF02) [**essentiel**]

4.3 Algorithmique

Des notions algorithmiques seront indispensables à la conception de compilateurs, et la manipulation de différentes structures de données doit être acquise et ne poser aucune difficulté.

En particulier, doivent être maîtrisé(e)s : piles, files, tables de hachage, arbres, graphes.

Modules concernés

— Algorithmique et combinatoire des structures discrètes (SLA4IF01) [**essentiel**]
— Analyse des algorithmes (SLA5IF09) [préférable]

4.4 Conception

Une partie de la conception du compilateur repose sur certains patrons de conception (*design patterns*). Il est souhaitable d'être à l'aise avec l'utilisation de ce type de méthodes. Pour autant, une bonne maîtrise de la programmation orientée objet peut permettre une assimilation rapide de ces pratiques.

Le patron principalement employé sera le *Visiteur*.

Modules concernés

— Conception orientée objet (SLA5IF02) [préférable]

4.5 Théorie des langages

La théorie des langages de programmation est centrale dans le cours de compilation. Pour autant, elle sera en grande partie considérée comme acquise, il y aura peu de rappels (et rapides).

En particulier, les notions suivantes doivent être parfaitement connues : reconnaissance d'un langage par un automate, grammaire formelle (symboles terminaux et non-terminaux). Représentation arborescente d'un langage...

Modules concernés

— Langages et automates (SLA5IF10) [**essentiel**]
— Langages algébriques [**essentiel**]

4.6 Systèmes d'exploitation

Pour comprendre la compilation d'un langage de haut niveau vers un langage assembleur, il est indispensable de maîtriser le fonctionnement des systèmes d'exploitation.

Modules concernés

— Systèmes d'exploitation (SMA7IF01) [important]

5 Contenu

— Analyse automatisée d'un langage de programmation (utilisation d'ANTLR4)
— Notions générales sur la compilation et l'interprétation.

— Explications de toutes les phases d'un compilateur, jusqu'au langage assembleur (MIPS32).
— Écriture en Java d'un compilateur pour un langage défini — en cours et en travaux pratiques
— Examen des différents concepts des langages de programmation, en fonction de ce qu'ils impliquent du côté du compilateur.

6 Objectifs

Lien entre programme de haut niveau et exécution machine À l'issue du cours, l'étudiant doit avoir une bonne compréhension théorique de toutes les étapes qui permettent de passer d'un langage de programmation à son exécution par le processeur.

Spécification et augmentation des langages L'étudiant doit être capable de comprendre les contraintes structurelles (liées aux langages machines et à l'architecture cible du compilateur) qui sous-tendent la conception d'un langage de programmation. Ce sujet concerne aussi l'extension de langages existant pour de nouvelles fonctionnalités : étendre un compilateur à un fragment de langage plus large demande de comprendre entièrement son fonctionnement.

Écriture d'un programme complexe Le projet réalisé pendant le semestre à travers les TP consistera en un programme structuré comportant de nombreux éléments reliés à des aspects nombreux et variés de l'informatique (de l'abstraction du langage à la génération de code machine).

7 Modalités d'évaluation

Le module est évalué en contrôle continu. Des devoirs sur table et/ou des TP notés auront lieu selon un agenda qui sera précisé au fil du semestre.

La construction du compilateur au fil des TP devra avoir été faite et comprise, pour une bonne réussite de ces évaluations.

8 Descriptif prévisionnel des séances de cours magistral

Première séance (2h)	Introduction (au tableau). Analyse lexicale et syntaxique (rappels).
Deuxième séance (1h30)	Présentation et commentaire d'un interpréteur rudimentaire écrit en OCaml. Visiteurs — Arbres de syntaxe abstraits
Troisième séance (1h30)	langages et types.
Quatrième séance (1h30)	Analyse sémantique — Table des symboles —
Cinquième séance (1h30)	Assembleur
Sixième séance (1h30)	Représentation intermédiaire.
Septième séance (1h30)	Génération de code
Huitième séance (1h30)	Graphes et algorithmes pour l'allocation de registres.
Neuvième séance (1h30)	Compilation de structures avancées, optimisations. Conclusions.

9 Références

Pour l'ensemble du cours *Modern compiler implementation in Java* de A.Appel est une référence complète.

9.1 Théorie des langages

Introduction à la théorie des langages de programmation de B. Meyer est une bonne référence. Je vous invite davantage à revenir aux ressources du cours de Langages Algébriques.