Langage de Programmation et Compilation

Jean-Cristophe Filliâtre

29 septembre 2023

Table des matières

Première partie

Cours 1 29/09

Introduction

Maîtriser les mécanismes de la compilation, transformation d'un langage dans un autre. Comprendre les aspects des langages de programmation.

0.1 Un Compilateur

Un compilateur est un traducteur d'un langage source vers un langage cible. Ici le langage cible sera l'asembleur.
Tous les langages ne sont pas compilés à l'avance, certains sont interprétés, transpilés puis interprétés, compilés à la volée, transpilés puis compilés... Un compilateur prend un programme P et le traduit en un programme Q de sorte que : $\forall P, \exists Q, \forall x,\ P(x) = Q(x)$. Un interpréteur effectue un travail simple mais le refait à chaque entrée, et donc est moins efficace.
Exemple : le langage *lilypond* va compiler un code source en fichier .pdf.

0.2 Le Bon et le Mauvais Compilateur

On juge un compilateur à :

1. Sa correction
2. L'efficacité du code qu'il produit
3. Son efficacité en tant que programme

   « Optimizing compilers are so difficult to get right that we dare say that no optimizing compiler is completely error-free ! Thus, the most important objective in writing a compiler is that it is correct »- *Dragon Book, 2006*

0.3 Le Travail d'un Compilateur

Le travail d'un compilateur se compose :

— d'une phase d'analyse qui :

1. reconnaît le programme à traduire et sa signification
2. signale les erreurs et peut donc échouer

— d'une phase de synthèse qui :

1. produit du langage cible
2. utilise de nombreux langages intermédiaires
3. n'échoue pas

Processus : source → analyse lexicale → suite de lexèmes (tokens) → analyse syntaxique → Arbre de syntaxe abstraite → analyse sémantique → syntaxe abstraite + table des symboles → production de code → langage assembleur → assembleur → langage machine → éditeur de liens → exécutable.

1 L'assembleur

1.1 Arithmétique des ordinateurs

On représente les entiers sur n bits numérotés de droite à gauche. Typiquement, n vaut 8, 16, 32 ou 64. On peut représenter des entiers non signés jusqu'à $2^n - 1$. On peut représenter les entiers en définissant b_{n-1} comme un bit de signe, on peut alors représenter $\left[-2^{n-1}, 2^{n-1} - 1\right]$. La valeur d'une suite de bits est alors : $-b_{n-1}2^{n-1} + \sum_{k=0}^{n-2} b_k 2^k$. On ne peut pas savoir si un entier est signé sans le contexte.
La machine fournit des opérations logiques (bit à bit), de décalage (ajout de bits 0 de poids fort, 0 de poids faible ou réplication du bit de signe pour interpréter une division), d'arithmétique (addition, soustraction, multiplication).

1.2 Architecture

Un ordinateur contient :

— Une unité de calcul (CPU) qui contient un petit nombre de registres et des capactités de calcul

— Une mémoire vive (RAM), composée d'un très grand nombre d'octets (8 bits), et des données et des instructions, indifférenciables sans contexte.

L'accès à la mémoire coûte cher : à 1B instructions/s, la lumière ne parcourt que 30cm entre deux instructions.