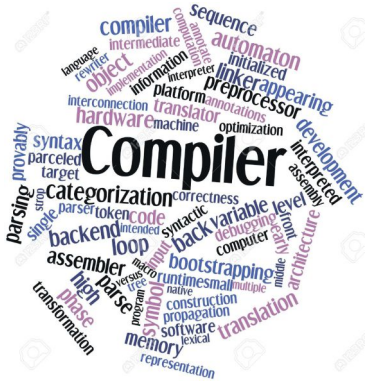
INTRODUCTION A LA COMPILATION

SUPPORT DE COURS COMPILATION
ESI-Compilation



Chebieb

Décembre 2020
V.0.03

Table des matières

Introduction

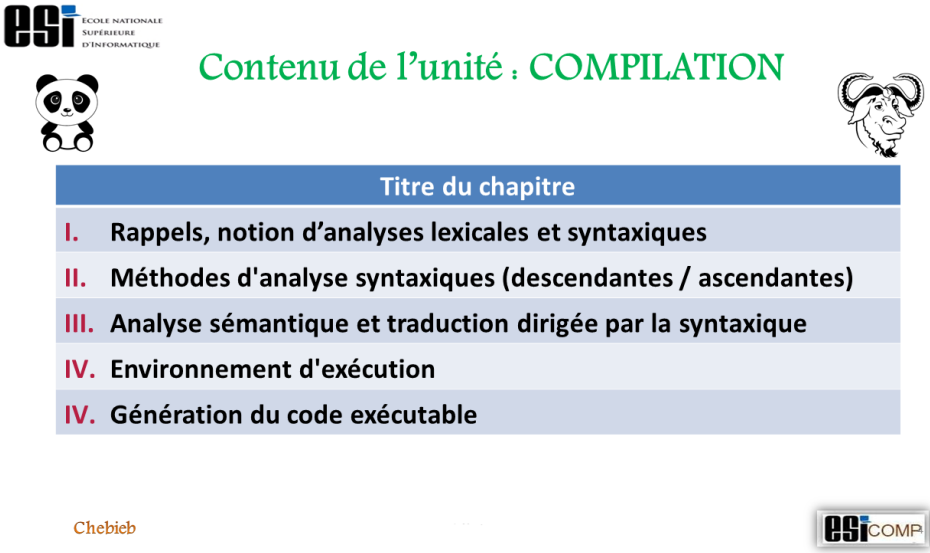
Dans ce cours nous présentant le contenue pédagogique COMPILATION dans l'objectif de comprendre comment nous allons aborder les notions essentielles de cette matière importante en informatique. Pour cela nous allons aborder les points suivants :

1. Présentation du programme à étudier , ses objectifs et les modalités d'évaluation des efforts d'atteinte de ces objectifs
2. Une première définition de la notion de compilation / compilateur

PRÉSENTATION DE L'UNITÉ COMPILATION

I

1. C'est quoi le programme ?



2. Objectis

COMPILATION

Objectifs

- Ecrire une grammaire d'un langage de programmation et construire un analyseur syntaxique pour ce langage à l'aide d'outils standard.
- Comprendre la description formalisée de la sémantique opérationnelle et de la sémantique statique d'un langage
- Programmer un compilateur d'un langage vers une machine cible

Chebieb

3. Prérequis

COMPILATION

Prérequis

- Théorie des langages de programmation et applications
- Programmation dans l'un des deux paradigmes (P. impérative, POO)
- Système d'exploitation (assembleur, registres du processeur...etc.)

Chebieb

4. Déroulement et évaluation

Structure de l'unité .COMPILATION.

- Unité à 4 crédits :
  - 1 Crédit = 15 heures en présentiel et **10 heures individuel**
  - 30 heures cours , 30 heures TD/TP , 40 heures individuel
- Cours : 2h/semaine
- TD / TP : 2h / semaine
- Travail individuel: 3 h/ semaine
- Contact : tpcompal@esi.dz

Chebieb

Modalités d'évaluation

- **Contrôle continu** : 15% sous plusieurs formes
  - TD : évaluée par des interros.
  - 3 TP au minimum
- **Travaux à remettre ou TP de synthèse (examen TP):** 15 %
- **Contrôle Intermédiaire : 20%**
- **Examen final : 50 %**

$$\text{NOTE FINALE} = CC * 0{,}15 + TP * 0{,}15 + CI * 0{,}2 + EF * 0{,}5$$

Chebieb

II

PREMIER PAS : C'est quoi un compilateur ? Expliquer la notion de compilation/compilateur et ses tâches.

Le cœur métier de l'informaticien est la programmation. C'est la première aptitude qu'un élève informaticien doit acquérir.

Plusieurs méthodes, outils et pratiques ont été créées pour rendre cette activité efficace et productive.

Le premier outil de base informatique dédié à l'activité de programmation est le COMPILATEUR. C'est un système informatique au cœur du processus de développement logiciel avec une très forte valeur ajoutée à même de faciliter la tâche de programmation (codage, implémentation).

L'étude des compilateurs complète la compréhension du fonctionnement de l'ordinateur et comment les développeurs puissent transmettre les instructions à la machine cible.

De plus cette étude permet de découvrir des principes et techniques applicables dans d'autres domaines dans la carrière de l'ingénieur informaticien.

1. L'évolution de la fonction de programmation

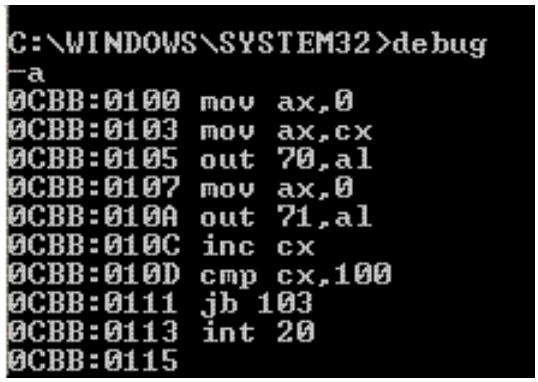
1.1. Du code machine à la programmation script

La programmation des premiers ordinateurs des années 40 s'effectuait par la codification des opérations de base (transfert de mémoire en registre, addition, comparaison....) directement en binaire "codes machine".

Exemple de carte perforée contenant les programmes pré-historiques

Carte perforée

Les années 50 ont vue l'utilisation de mnémoniques (représentation symboliques des codes machine) qui pouvait être regroupées en macro-instructions (des raccourcis pour nommé un bloc qui est ré-utilisable). Exemple : Assembleur 8086 INTEL / 68009 MOTORAL / SPARC de SOLARIS / ADSP



Exemple de programme Assembleur pour les machines INTEL sous Windows

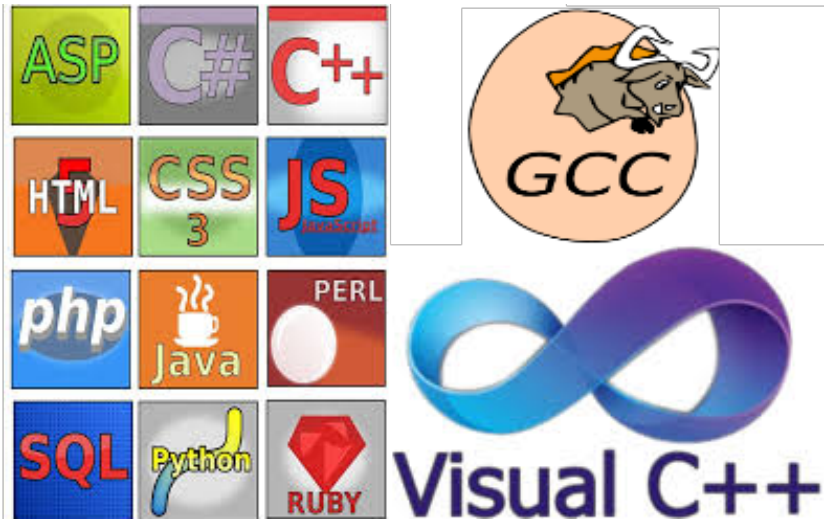
Assembleur Intel/Windows

Depuis la fin des année 50 début des années 60s, sont apparus les langages de programmation tels que Fortran, COBOL, C, C++, C#, LISP, Python, JAVA, REACT etc. Les programmes écrits dans ces langages contiennent peut de détails du langage machine cible. Ci-dessous un exemple d'un programme écrit en C :

```c
#include <stdio.h>
int a,b,p,i;
 void main (int argc, char *argv[])
{
   a = argv[1];
   b = argv[2];
   if (a==0 || b==0)
         p=0;
     else {
         p=a;
         if (a<b){
              p=b;
              b=a;
```

```
14                a=p;
15              }
16          for (i=1;i<b;i++)
17              p=p+a;}
18 }
```

Chaque évolution vers un palier éloigné de la machine (processeur) est réalisée grâce à un système (un logiciel) spécifique nommé **COMPILATEUR** dont la fonction est de transformer les instructions écrites dans un niveau en instructions du niveau le plus bas. L'image ci-après montre le logos de compilateurs les plus utilisés ...entre autre GCC, Microsoft Visual, Borland, JAVAC, PHP



La réussite spectaculaire et parfaite de cette catégorie de systèmes informatiques (logiciel) est la parfaite illustration de **l'utilité des outils d'abstraction mathématique pour résoudre les problèmes complexes d'une façon élégante** à savoir : la théorie des langages, la théorie des graphes et les techniques de programmation linéaire et optimisation.

2. Définition de la compilation et du compilateur

2.1. Compilation

Rappel : En littérature (Larousse)

Tirer des extraits de divers textes et les réunir pour faire un ouvrage de seconde main.

Définition : Compilation

Le processus qui consiste à transformer un programme écrit dans un langage de haut niveau (dit code source lisible par un humain) en un fichier binaire exécutable par une machine (dit code objet).

Les étapes de ce processus sont réalisées par un programme dit **compilateur** (ou traducteur)

2.2. Compilateur

Définition : Compilateur

Le programme qui transforme (on dit traduit aussi) un programme source dit **"code source"** écrit dans un langage de haut niveau vers un langage de bas niveau dit **"code objet"** (ex langage machine) on dit aussi un traducteur.

2.3. Cas particuliers de compilateurs

Définition : L'assembleur

- Transforme le programme source écrit sous forme d'une représentation symbolique des instructions machine dites "mnémoniques" (ex : assembleur MASM) vers le langage machine exécutable.

Définition : Le pré-processeur

- Traduit des directives (méta-instructions) écrites dans un langage de haut niveau vers des commandes ou des instructions au compilateur (ex : préprocesseur JAVA qui traduit la directive **import** en une action du compilateur JAVAC qui consiste à inclure une référence externe au programme.

Définition : L'interpréteur

- Traduit le programme source écrit dans un langage de haut niveau et les exécute immédiatement sur leur données sans générer leur code machine (ex : le compilateur Visual Basic / VBA est un interpréteur)

Définition : Le transpileur (transpiler) :

- Traduit le programme source d'un langage de programmation et le compile dans un autre langage de programmation. C'est un compilateur source-à-source écrits dans deux langages avec approximativement le même haut niveau. (ex : Babel compile du JAVASCRIPT pour les navigateurs ; HPHPc qui transforme du code PHP en C++

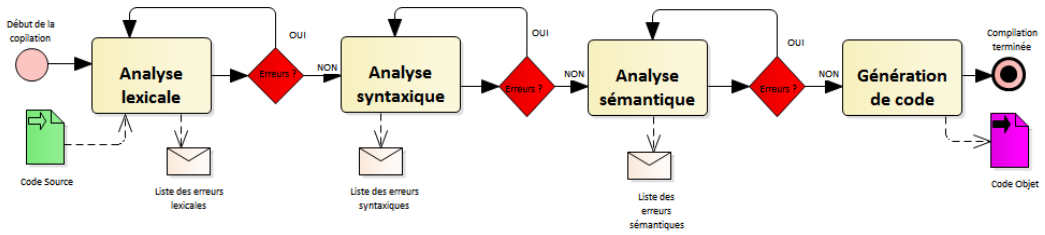
3. La chaîne de compilation

3.1. Étapes essentielles de la compilation

Définition : Chaîne de compilation

La chaîne de compilation est l'ensemble générique des étapes importantes que subit le code source durant le processus de transformation vers le code objet cible.

Chaque compilateur implémente ces étapes selon son type et son architecture (sa structure). La figure ci après illustre le processus de compilation.

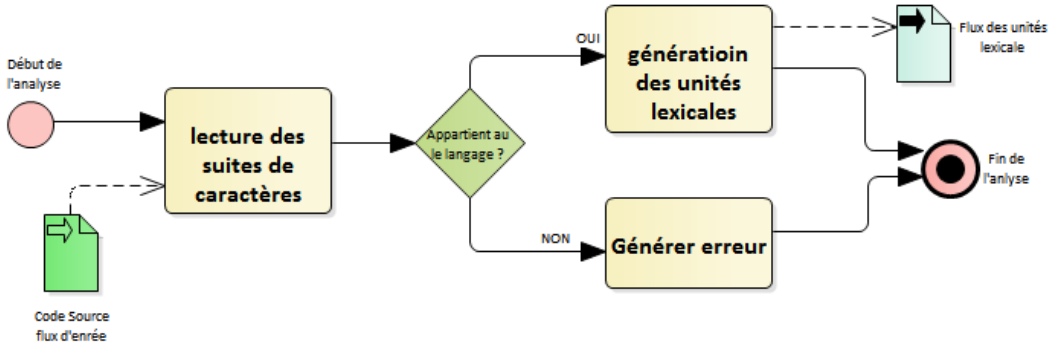


Étapes de Compilation

3.2. Analyse lexicale

Définition : Analyse lexicale

Étape qui consiste à lire le programme source dit **flux d'entrée** (des suites de caractères) et le traduire en **unités lexicales dites lexèmes** (*Token*) . Un message d'erreur est généré lorsque une suite de caractères dans le flux ne correspond pas au langage source. La figure ci-après illustre ce processus.



Analyse lexicale

Définition : Unité lexicale

Une suite de caractères ayant une signification en terme de mots du langage de programmation utilisé par le code source. Le lexème est un mot du langage source. Un identifiant, un mot clé, un opérateur...

Chaque unité lexicale possède un code et une valeur qui pointe vers une entrée dans une structure de donnée dite **table des symboles** qui stocke son type et la suite de caractères lue. Certaines unités peuvent ne pas avoir d'entrée dans la table.

Le code correspondant à une catégorie de mot dans le langage (un mot-clé, identificateur, constante)

Exemple : Un flux en entrée représentant une instruction JAVA

if (a > b) a= a+1 ; l'analyseur identifie les unités lexicales suivantes :

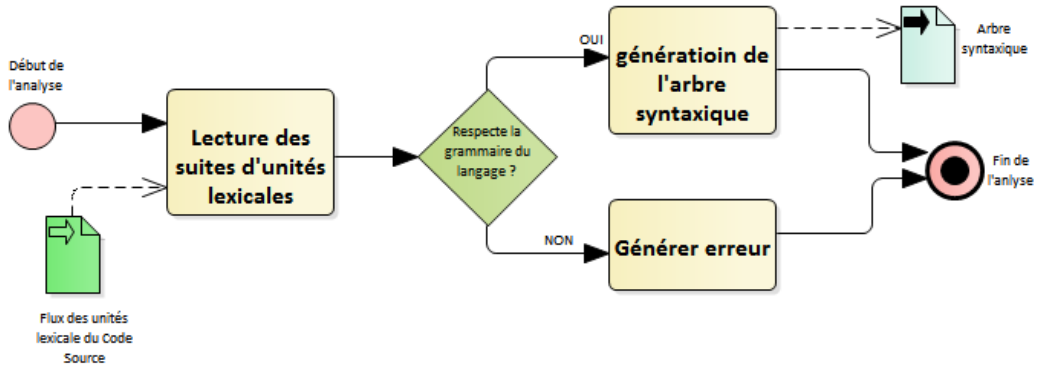
keyword_if (identifiant.340 > identifiant .366) identifiant.340 = identifiant.340 + constante.330

keyword_if : n'est pas dans la table des symboles

3.3. Analyse syntaxique

Définition : Analyse syntaxique

Consiste à vérifier si les unités lexicales obtenues du code source forme bien les structures du langage utilisé. C'est à dire est ce que **la grammaire du langage** utilisé est respectée par le flux des lexème du code source ? si oui un structure de donnée dite "**ARBRE SYNTAXIQUE**" est générée sinon une erreur est signalée. La figure ci-après montre les Phases de l'analyse syntaxique.



Phase analyse syntaxique

*Exemple : Pour le flux : A + B *C*

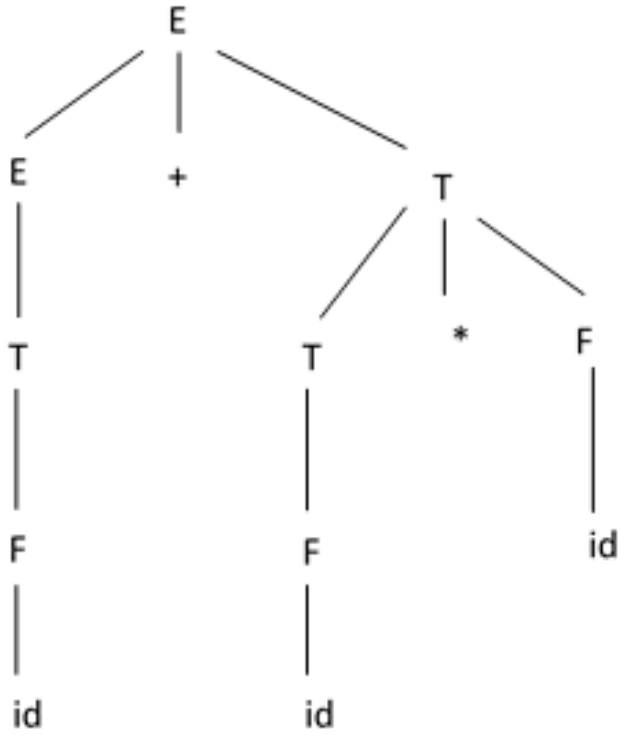
Soit la grammaire du langage :

E → E + T | T

T → T * F | F

F → (E) | id

L'analyse lexicale retournera par exemple : id1 + id2 * id3 et l'analyse syntaxique retournera l'arbre syntaxique de cette chaîne.



Arbre syntaxique

3.4. Analyse sémantique

Définition : Analyse sémantique

Dans cette phase des tâches spécifiques sont exécutées pour des considérations diverses spécifiques au langage considéré. Par exemple le contrôle de type des opérandes d'une opération ou vérifier que les indices d'un tableau sont bien des entiers.

Ces actions sont dites **règles sémantiques** ou **actions sémantiques**.

3.5. Génération de code

Définition : Génération de code

C'est la phase qui consiste à générer les séquences d'instructions dans le langage objet. Elle est constituée de 3 étapes :

- **Génération du code intermédiaire** : générer un code sous une forme intermédiaire facile à traduire dans le langage cible. Une de ces formes est le code à trois adresses prenant en compte la forme des instructions machine cible
- **Optimisation** : une étape qui consiste à améliorer le code intermédiaire pour réduire le temps d'exécution et/ou l'espace mémoire qui sera occupé par le programme final
- **Génération du code objet** : Étape finale qui consiste à convertir le code intermédiaire optimisé en instruction de la machine cible.

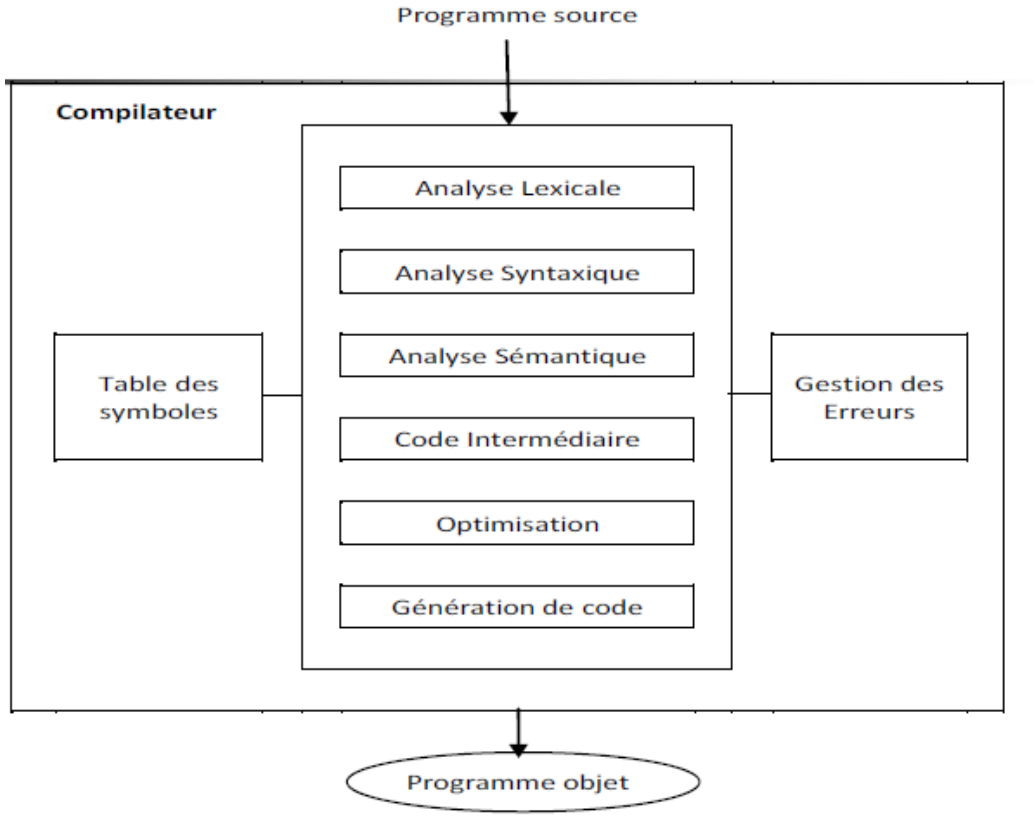
Cette phase de génération exploite aussi la table des symboles créée lors de l'analyse lexicale.

4. Structures du compilateur

4.1. Composants d'un compilateur

Le compilateur possède un module réalisant chaque étapes de la compilation.

Les structures des compilateurs diffèrent selon leur technologie et leur utilisation. La figure ci- après montre la structure générique du compilateur.



Composants d'un compilateur

4.2. Outils pour le développement des compilateurs

Méthode : Générateurs d'analyseurs et de compilateurs

Il existe des programmes utilisés dans la constructions des compilateurs :

- Générateurs d'analyseurs lexicaux : exemple LEX
- Générateurs d'analyseurs syntaxiques : exemple YACC (Yet Another CompilerCompiler)
- Générateurs de compilateurs : c'est des outils qui génèrent pour une grammaire donnée un analyseur lexical et syntaxique (ANTLR, COCO/R, JAVACC)

* *

*

La compilation est une activité au cœur du développement logiciel quelque soit leur complexité (des petits programmes aux systèmes complexes). Elle est au cœur même de l'exploitation des ordinateurs (de tout types d' architecture). Elle consiste à transformer (traduire) un programme écrit dans un langage de programmation (code

source) vers un autre langage (code cible ou objet). Dans la plus part des cas le langage source est dit de haut niveau (langage symbolique compris par les programmeurs humains) et le langage cible est dit code machine soit en binaire, soit en assembleur. L'assembleur étant un langage où le instructions machine sont écrite sous forme de mnémoniques.

Le compilateur est le logiciel qui permet d'automatiser cette transformation. Les compilateurs sont de plusieurs types selon la manière dont ils réalisent la transformation (la traduction): les interpréteurs traduisent et exécutent le programme source ; les compilateurs traduisent et génèrent un code objet sans l'exécuter ; les pré-processeurs traduisent le code source en des instructions au compilateur lui même qui aident à compléter le code source afin qu'ils soit pré à la compilation ; les *"transpileurs"* (en anglais **transpilers**) traduisent un code source d'un langage de haut niveau vers un langage de haut niveau (PhP vers du C++).

Les étapes de compilation sont : l'analyse lexicale (lecture du code source et formation des unités lexicale) ; l'analyse syntaxique (lecture des unités lexicale et formation de l'arbre syntaxique) ; l'analyse sémantique (exécution d'actions spécifiques au langages tel que le contrôle de type) ; l'optimisation et la génération du code (génération d'un code intermédiaire à 3 adresses facile à optimiser et sa transformation en un code dans le langage cible).

Nous ne considérons pas l'édition des liens effectuée par le **lieur** comme une étape de compilation. C'est une étape dans la construction du programme final exécutable sur la machine cible ; cette étape est désignée par la fonctionnalité **"BUILD"** dans la majorité environnements de développement (IDE).

La compréhension de la structure et du fonctionnement des compilateurs nous permet de voir plusieurs techniques utilisant des outils mathématiques très puissants permettant de résoudre d'une façon relativement simple des problèmes complexes que nous pouvons rencontrer dans le domaine de l'ingénierie informatique.

Bibliographie

Ait Aoudia S. *Compilation : cours et exercices corrigés* OPU Alger 2014

Cooper,K., Torczon, L. *Engineering a compiler* Elsevier 2011

Aho. A.V.,& Ullman, J.D. *The theory of parsing, translation, and compiling* Prentice-Hall, Inc. 1972

Aho. A.V.,& Ullman, J.D. *Principles of compiler design translation (Vol. 21)* Reading, Mass. : Addison-Wesley. 1977

Aho. A.V., Sethi,R. & Ullman, J.D. *Compilers: Principles, Techniques, and Tools* Addison-Wesley 1986

Aho. A.V., Lam, M.S., Sethi,R. & Ullman, J.D. *Compilers: Principles, Techniques, and Tools* Pearson Education, Inc. 2006