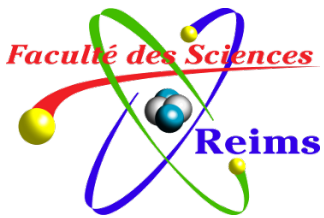
Introduction

Cyril Rabat
cyril.rabat@univ-reims.fr

Licence 3 Informatique - Info0602 - Langages et compilation

2019-2020



Cours n°1

Qu'est qu'un compilateur ?

Les phases de la compilation

Version 17 décembre 2019

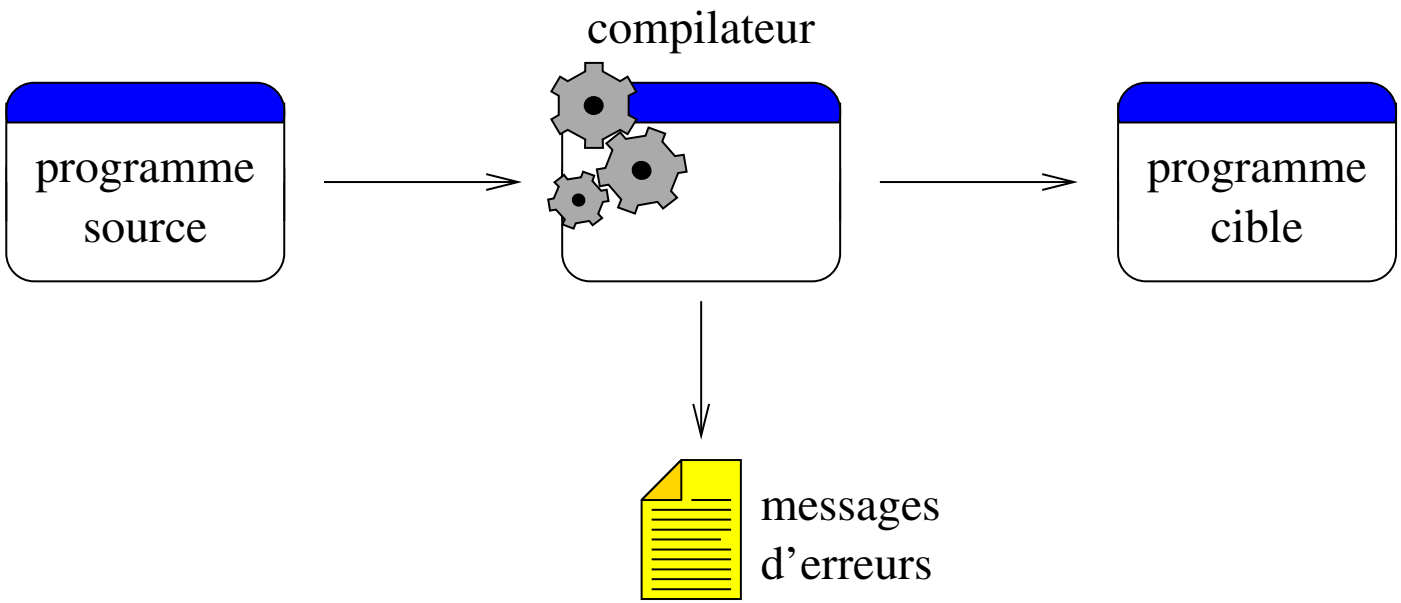
Table des matières

Table des matières

Un compilateur

Définition

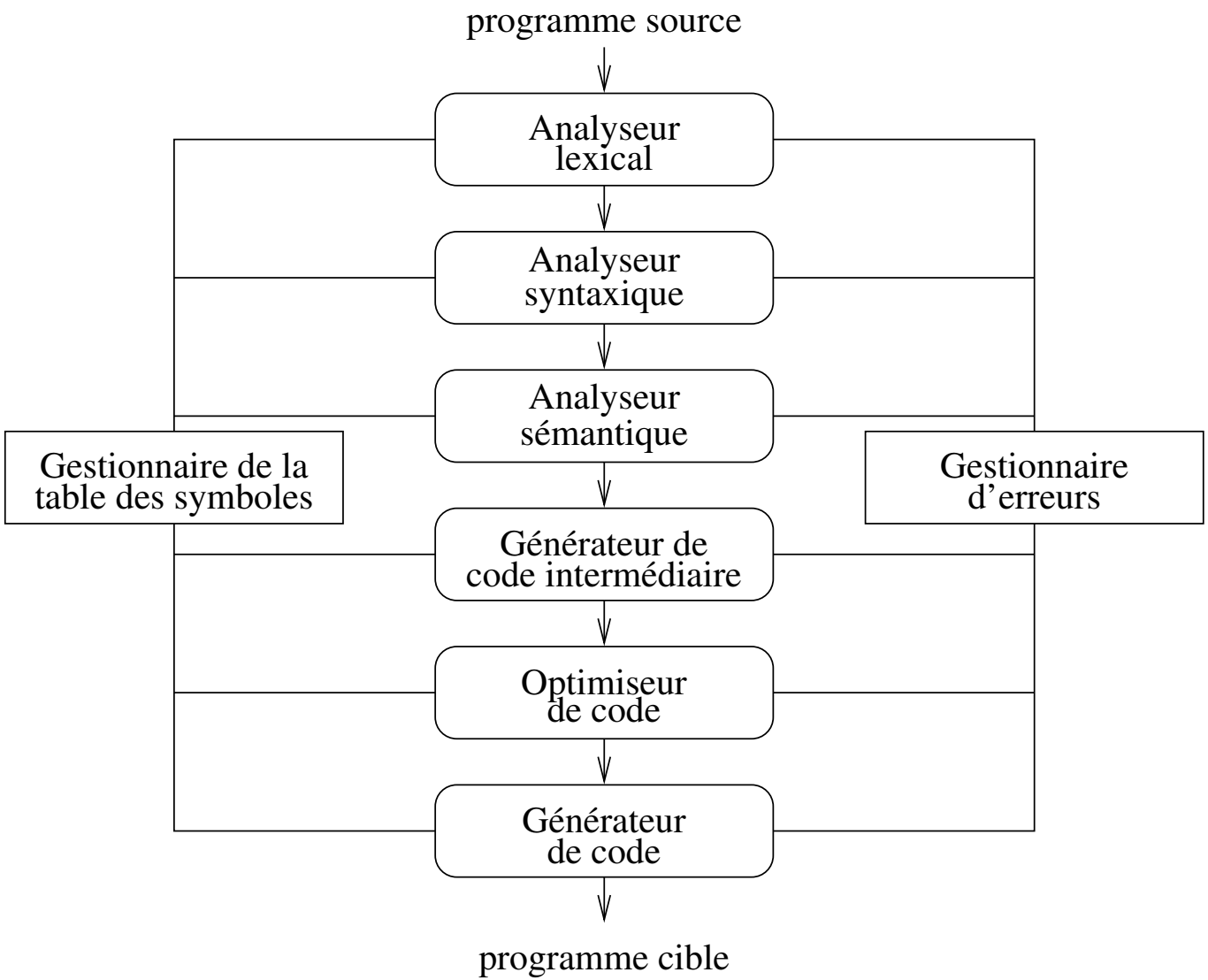
Programme qui lit un programme écrit dans un premier langage (le **langage source**) et le traduit en un programme équivalent écrit dans un autre langage (le **langage cible**).



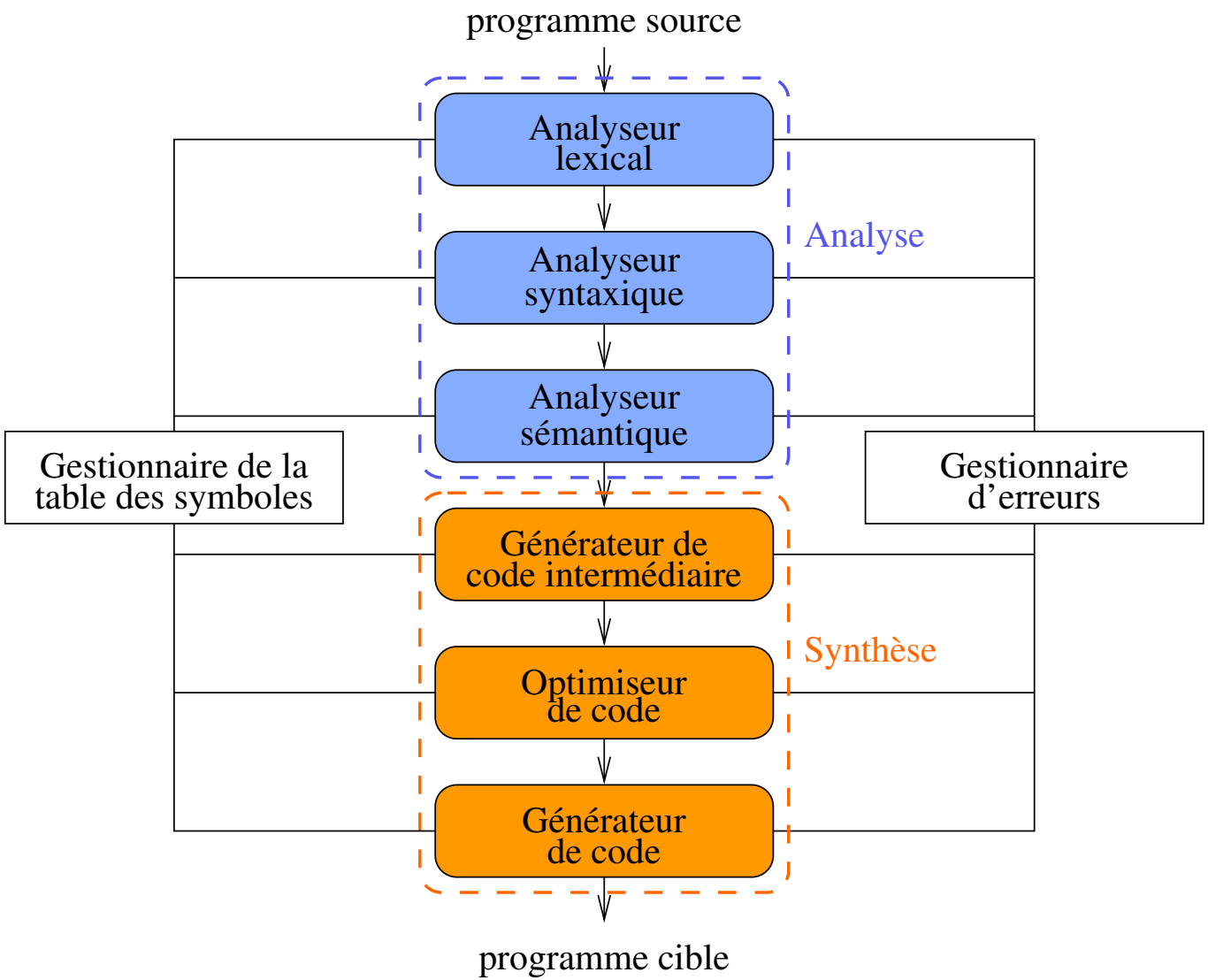
Les concepts

- La compilation fait appel à :
  - La théorie des langages
  - L'architecture des machines
  - L'algorithmique et le génie logiciel
- Concepts utilisés dans bien d'autres domaines :
  - Traitement de texte
  - Interpréteur de commandes / requêtes

Les phases d'un compilateur



Les phases d'un compilateur



Les deux parties de la compilation

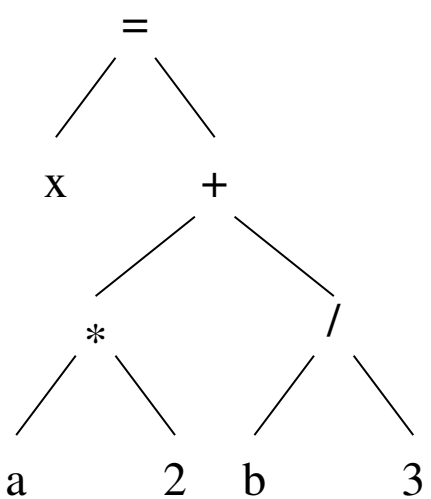
- **Analyse** :
  - Partitionnement du programme source en constituants
  - Création d'une représentation intermédiaire
- **Synthèse** :
  - Construction du programme cible à partir de la représentation intermédiaire

La représentation intermédiaire

- Pendant l'analyse, récupération et conservation des opérations du programme source
- Utilisation d'une structure hiérarchique : un arbre
- Exemple d'arbre utilisé : un arbre abstrait

Exemple

- Instruction : $x = a \times 2 + b/3$



Environnement du compilateur

- Création du programme source :
  ↪ Compilateur + autres programmes
- Nécessité d'assembler différentes sources :
  - Plusieurs fichiers sources
  - Utilisation de macros, etc.
    ↪ Réalisé par le préprocesseur

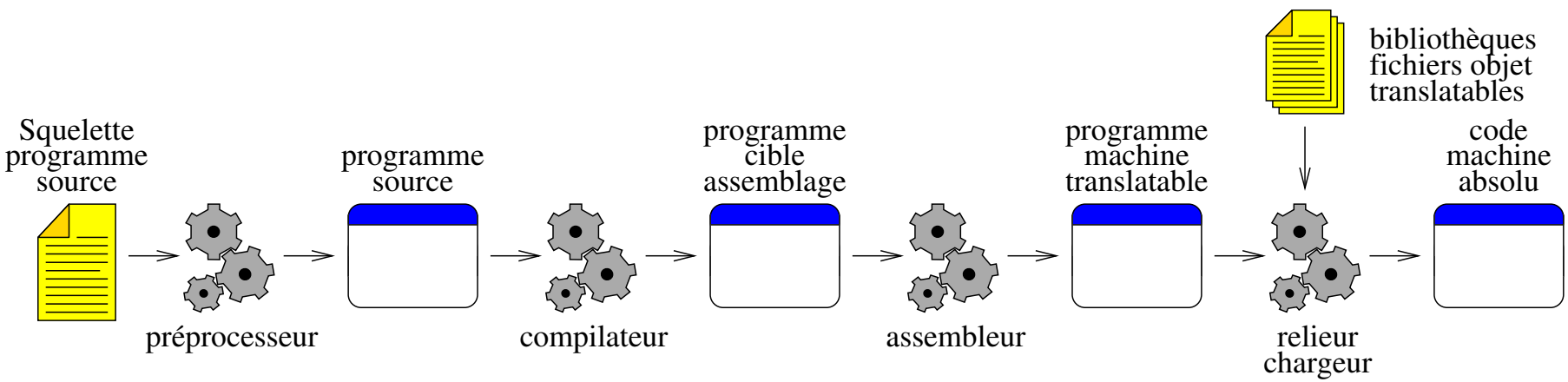


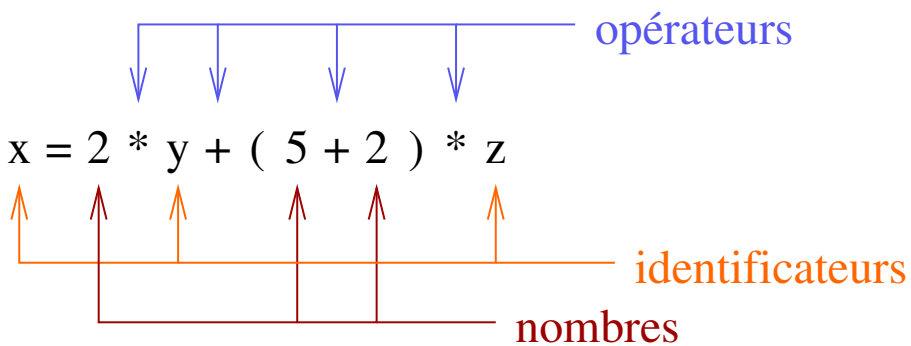
Table des matières

Les différentes phases de l'analyse

1. Analyse linéaire ou **analyse lexicale**
2. Analyse hiérarchique ou **syntaxique**
3. Analyse sémantique

Analyse lexicale

- But : reconnaître les **unités lexicales** (ou **lexèmes**)
- Exemples :
  - Identificateurs et mots-clés du langage
  - Les opérateurs
  - Les valeurs

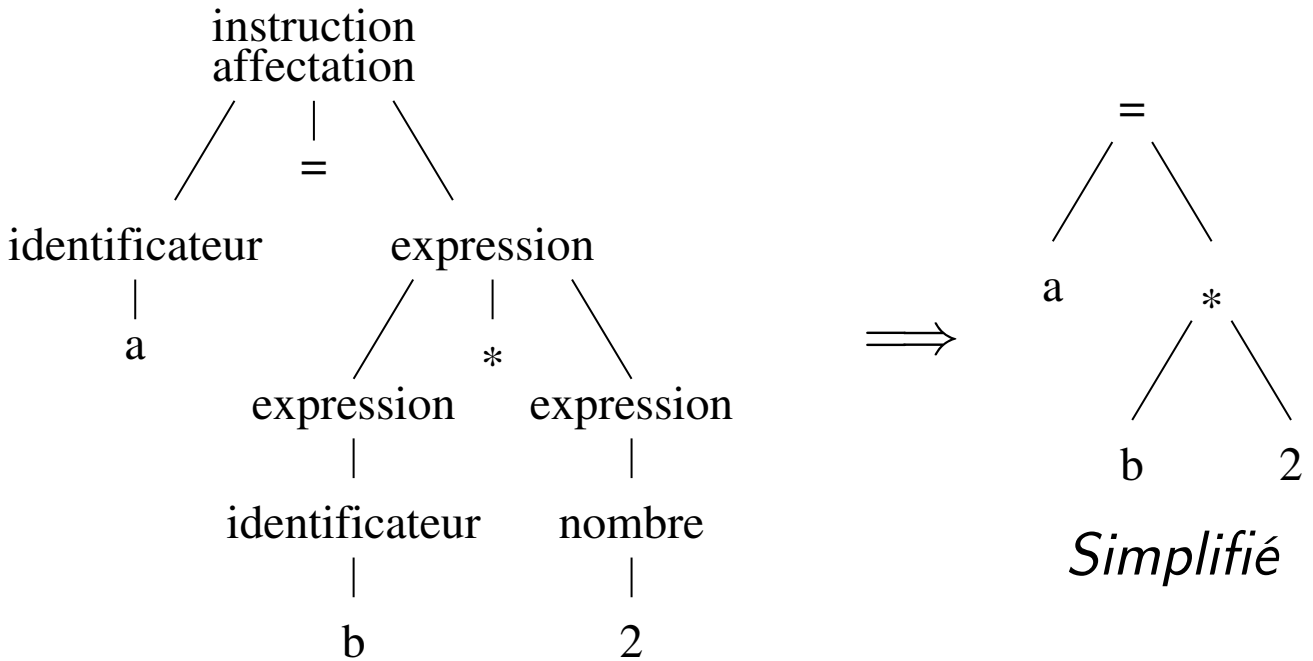


Analyse syntaxique (1/2)

- Appelée aussi **analyse grammaticale**
- But : regrouper les unités lexicales en structures grammaticales
- Généralement, utilisation d'arbres syntaxiques
- Structure hiérarchique généralement exprimée par des règles récursives
  $\hookrightarrow$ Utilisation de **grammaires**

Analyse syntaxique (2/2)

- Exemple de règles pour une expression :
  1. Tout *identificateur* est une expression
  2. Tout *nombre* est une expression
  3. Si exp_1 et exp_2 sont des expressions, il en est de même pour :
     - $exp_1 + exp_2$
     - exp_1 * exp_2
     - (exp_1)
- Exemple d'arbre syntaxique de $a = b * 2$:



Analyse sémantique

- But : vérifie si le programme source contient des erreurs sémantiques
  ↪ Exemple : la vérification des types
- Exploite l'arbre syntaxique
- Exemple avec l'instruction précédente, si *a* est un réel :

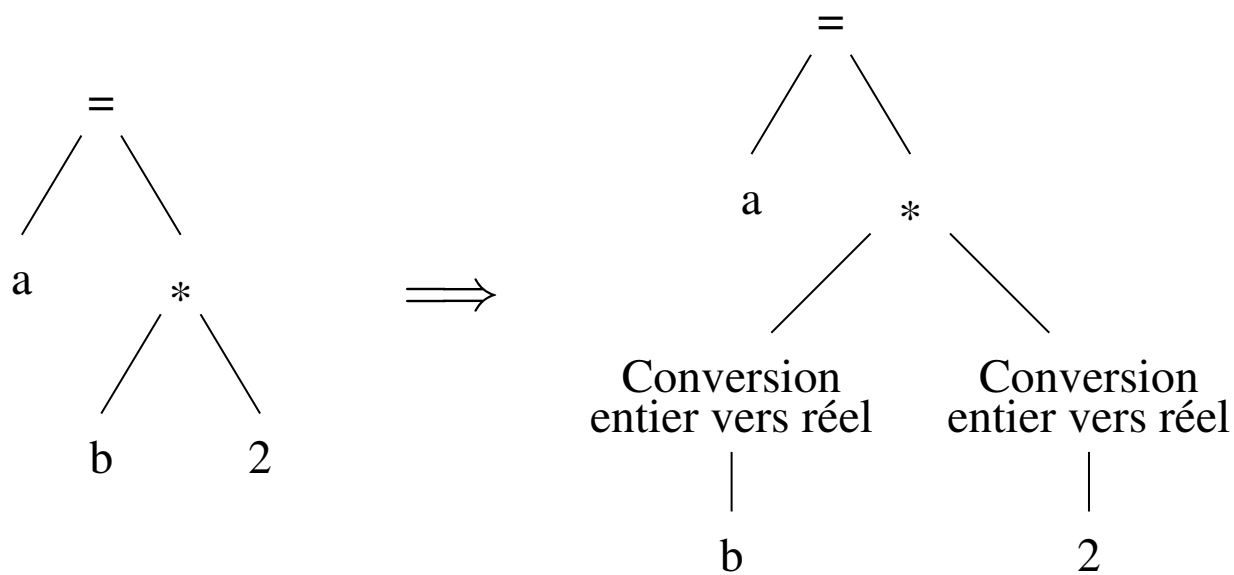


Table des matières

Les différentes phases de l'analyse

- La génération de code intermédiaire
- L'optimisation de code
- La génération de code

La génération de code intermédiaire

- But : construire une représentation intermédiaire du programme source
- Deux propriétés pour cette représentation :
  - Facile à produire
  - Facile à traduire
- Exemple : le code à 3 adresses
  $\hookrightarrow$ Séquence d'instructions dont chacune a au plus 3 opérandes
- Nécessite :
  - D'utiliser des variables intermédiaires
  - De faire un choix sur l'ordre de calcul
- Exemple : $x = y + 2 * z$ (avec id1 = x, id2 = y et id3 = z)

```
tmp1 = 2
tmp2 = id3 * tmp1
tmp3 = id2 + tmp2
id1 = tmp3
```

L'optimisation de code

- But : améliorer le code intermédiaire
- Réduction du nombre de variables
- Réduction du nombre d'instructions
- L'une des phases les plus gourmandes en temps
- Exemple :

```
tmp1 = id3 * 2
id1 = id2 + tmp1
```

La génération de code

- But : produire du code machine translatable ou du code en langage d'assemblage
- Choix des emplacements mémoires pour les variables
- Traduction des instructions du code intermédiaire en instructions machine
  ↪ Nécessite d'assigner les variables aux registres
- Exemple :

```
MOVF id3, R2
MULF #2.0, R2
MOVF id2, R1
ADDF R2, R1
MOVF R1, id1
```

Table des matières

La table des symboles

- Lors de la compilation, nécessité de collecter les informations sur les identificateurs
  ↪ Attributs tels que le type, la valeur, l'emplacement mémoire, la portée
- Table de symboles : structure de données contenant un champ pour chaque identificateur
- Création des champs lors de l'analyse lexicale
- Ajout des attributs lors des phases suivantes
- Permet aux différentes phases de trouver les informations nécessaires
  ↪ Exemple : le type lors de l'analyse sémantique

La gestion des erreurs

- Génération d'erreurs lors de chaque phase
- Objectif : traiter l'erreur pour poursuivre autant que possible
  $\hookrightarrow$ Éviter l'arrêt dès la première erreur !
- Exemple d'erreurs :
  - Analyse lexicale : flot de caractères non reconnu
  - Analyse syntaxique : construction incorrecte
  - Analyse sémantique : type incorrect