



ALF

Introduction

# L'équipe

---

- Cours:
  - Alexandru Radovici
- TP:
  - Amalia Simion
  - Adrian Popescu
- Devoir:
  - Amalia Simion
- Ressources
  - Bogdan Nițulescu et l'équipe de CPL de ACS

# Règles du cours

---

- Nous vous conseillons de venir au cours
- Si vous venez en classe, vous devez respecter ces règles
  - soyez attentif et **posez le plus de questions possible**

# Les cours de programmation

---

LP

POO

SDA

ALF

AO

**Keith Cooper, Linda Torczon, *Engineering a Compiler***

**Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools, 2<sup>nd</sup> Edition***

**Terence Parr, *The Definitive ANTLR 4 Reference*,  
(2nd Edition)**

# Ressources pour le cours ALF

---

- Site web: <https://upb-fils-alf.github.io/>
- GitHub issues: <https://github.com/UPB-FILS-ALF/questions/issues>
- Diapositives de cours
- La biobliographie

## Cours

- 12 cours
- diapositives
- bibliographie
  - Très important de lire



Kotlin



ANTLR

## TP

- 7 TP
- Programmation en Kotlin, ANTLR
- C'est important de collaborer avec votre collègues

## Contenu

- Introduction en Kotlin
- Analyse syntactique
- Analyse sémantique
- Génération de code

## Développent

- 8-20 heures par une devoir
  - Test des devoirs avec github classroom
- Questions sur Github Issues

**Les devoirs sont individuelles**



- diviser les chaînes en données significatives
- comment fonctionne le langage du processeur
- comment écrire votre propre langage de programmation

# Vous utiliserez ces pour

---

- Vérification des données
- Extraction des information dans des fichier
- Faire de langages de programmation
- Utilisez le langage d'assemblage
- Apprendre plus vite un langage de programmation
- Un vue d'ensable sure un system avec des ordinateurs

- Semestre
  - Tests de cours (2p)
  - Tests de TP (3p)
  - 4 devoirs (8p)
- Session d'examen
  - épreuve écrite (2p)
- Note
  - > 4.5 passer
  - présence TP

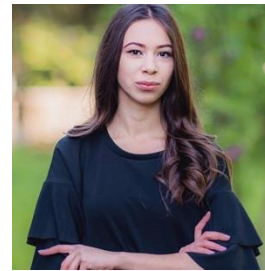
# Hall of Fame



Mihai Costin  
2023



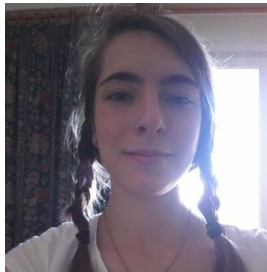
Daniela Constantin  
2019



Diana Ghindăoanu  
2019



Teodor Deaconu  
2018



Catrina Bodean  
2018



Claudia Dumitru  
2017



Catalin Stancu  
2017



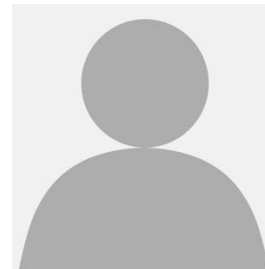
Student



Student



Student



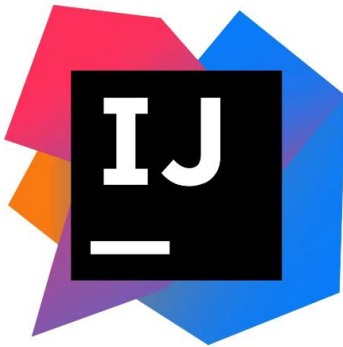
Student



Student

# Outils logiciels recommandés

---



IntelliJ Community Edition



GitHub Codespace (VS Code)



- Britannique
- Mathématicien
- Machine de Turing
  - Équivalent à des ordinateurs modernes

- Informatique
- Quelques mots sur ALF
- Compilateurs
- Sujets

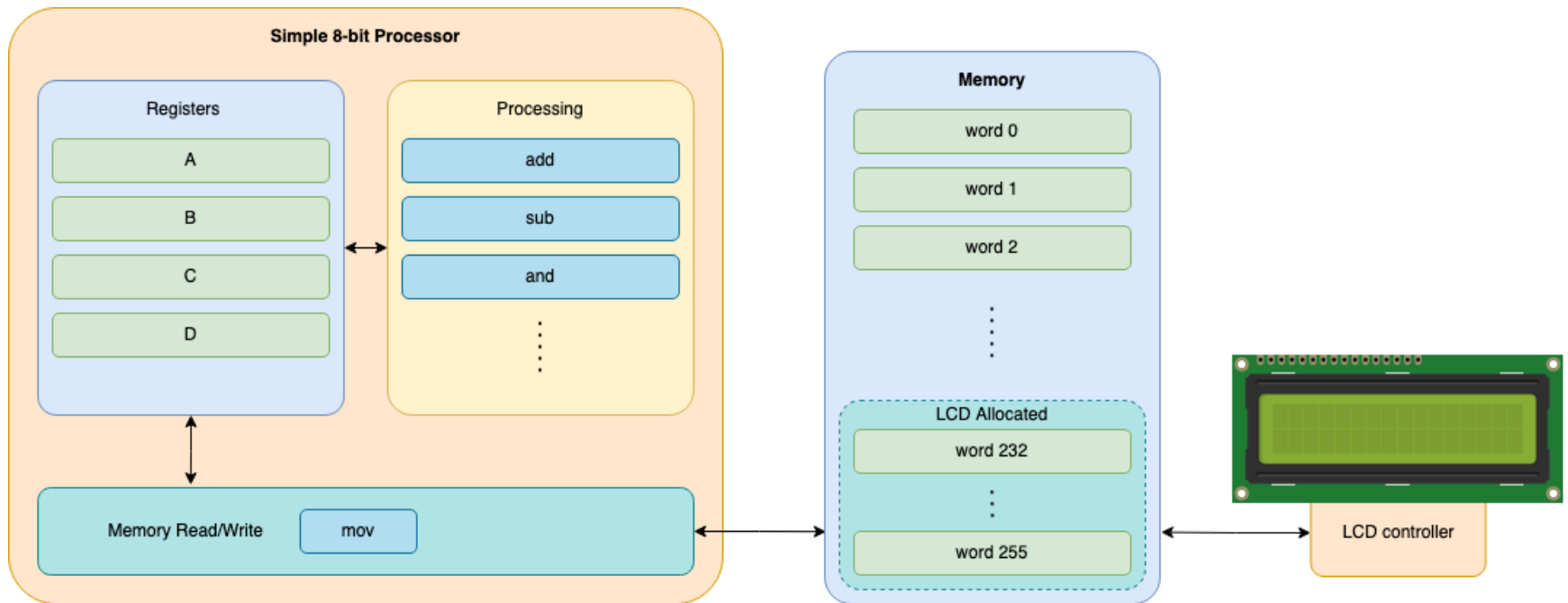


- Engineering a Compiler
  - Chapitre 1
- Compilers: Principles Techniques and Tools
  - Chapitre 1
    - 1.1
    - 1.2



# Système Informatique

Simulateur <https://schweigi.github.io/assembler-simulator/>



# Langage haute niveau (Java)

---

```
class Main {  
    public static void main (String args[])  
    {  
        LCD.write ("Hello World!");  
    }  
}
```

# Langage haute niveau (Kotlin)

---

```
fun main() {  
    LCD.write("Hello World")  
}
```

# Assembleur

; Simple example

; Writes Hello World to the output

JMP start

hello:

DB "Hello World!" ; Variable

DB 0 ; String terminator

start:

MOV C, hello ; Point to var

MOV D, 232 ; Point to output

CALL print

HLT ; Stop execution

print: ; print(C:\*from, D:\*to)

PUSH A

PUSH B

MOV B, 0

.loop:

MOV A, [C] ; Get char from var

MOV [D], A ; Write to output

INC C

INC D

CMP B, [C] ; Check if end

JNZ .loop ; jump if not

POP B

POP A

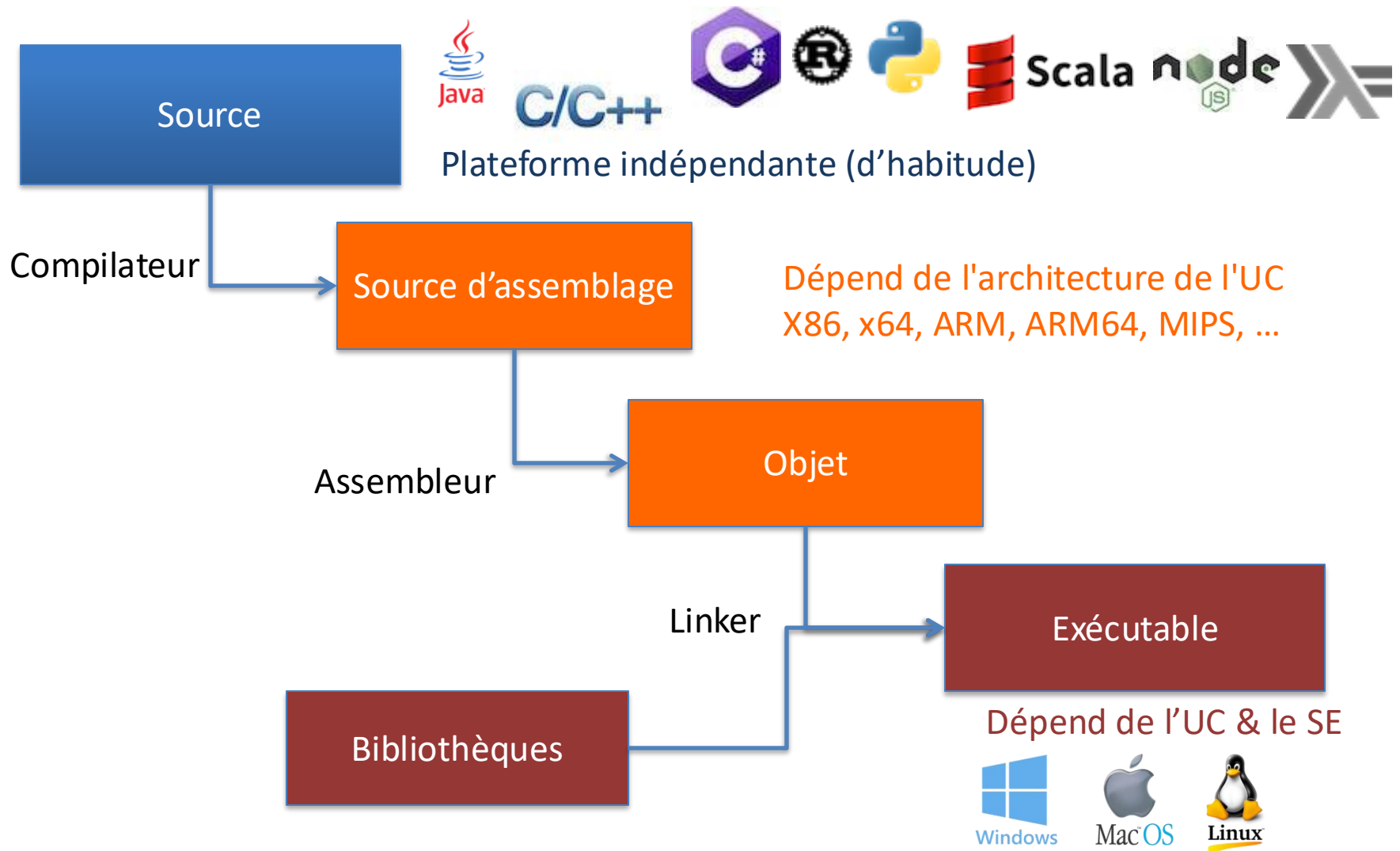
RET

# Binaire

1F	0F	48	65	6C	6C	6F	20	57	6F	72	6C	64	21	00	06
02	02	06	03	E8	38	18	00	32	00	32	01	06	01	00	03
00	02	05	03	00	12	02	12	03	15	01	02	27	1F	36	01
36	00	39	00	00	00	00	00	00	00	00	00	00	00	00	00

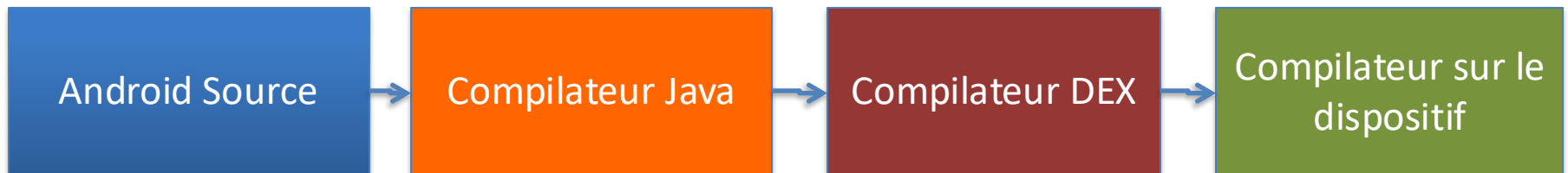
- Interprété
  - PHP
- Compilé
  - C/C++, Rust
- Interprété et compilé (JIT)
  - Java, Kotlin, Python, NodeJS

# Compilateur



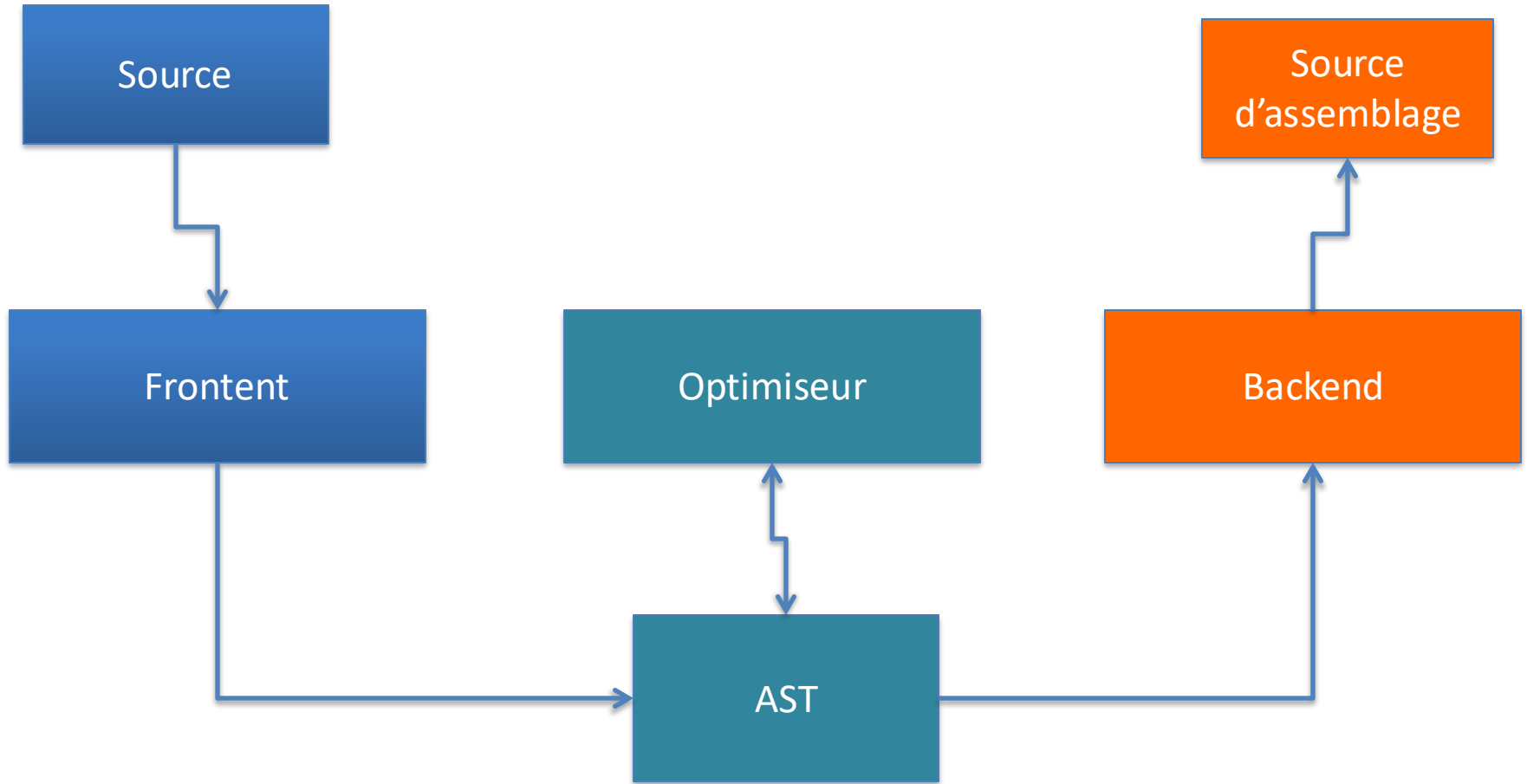
# Les compilateurs se trouve

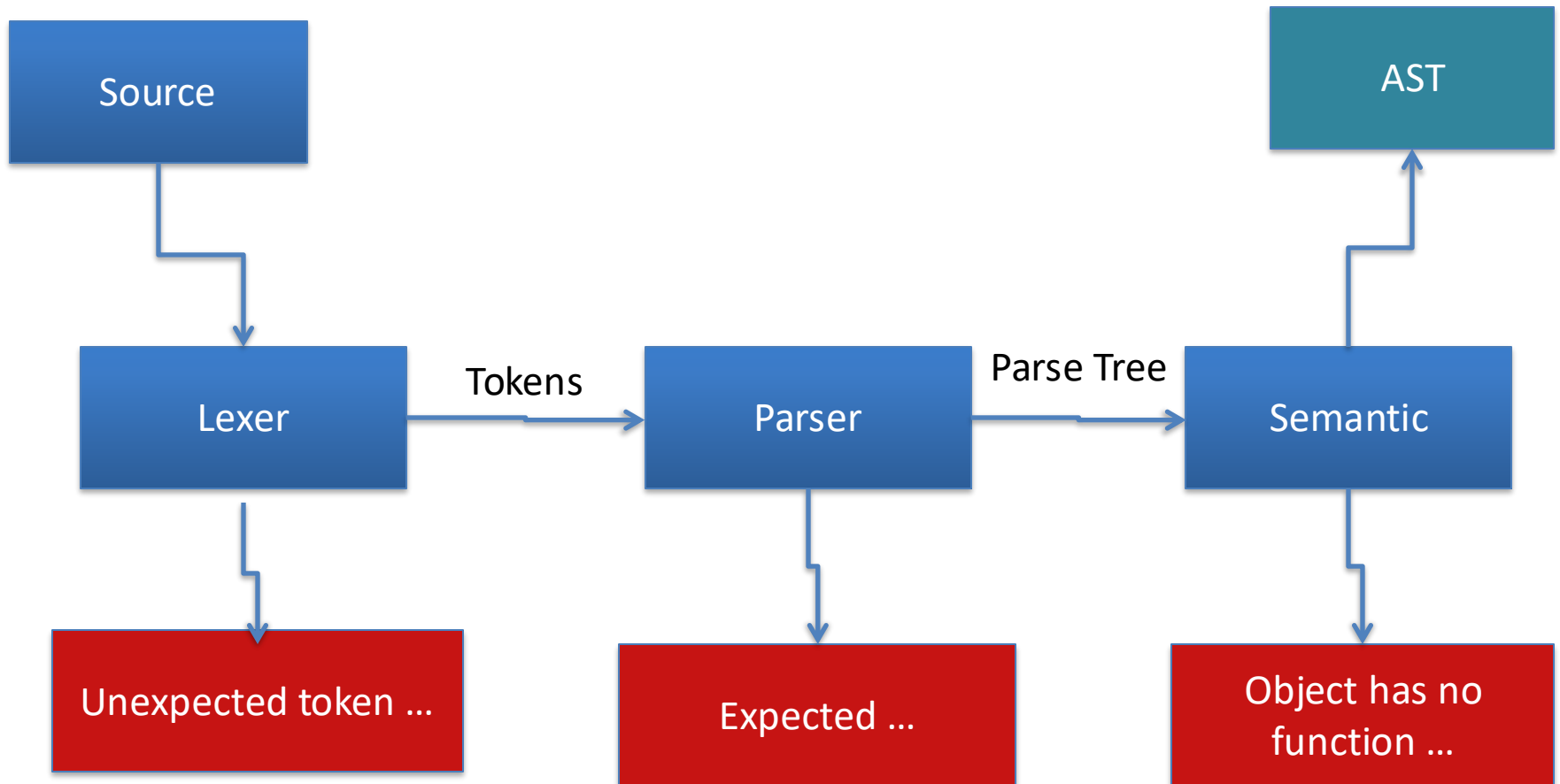
- SDK de langages
- Pilot de cartes vidéo
- Android Runtime (ART)



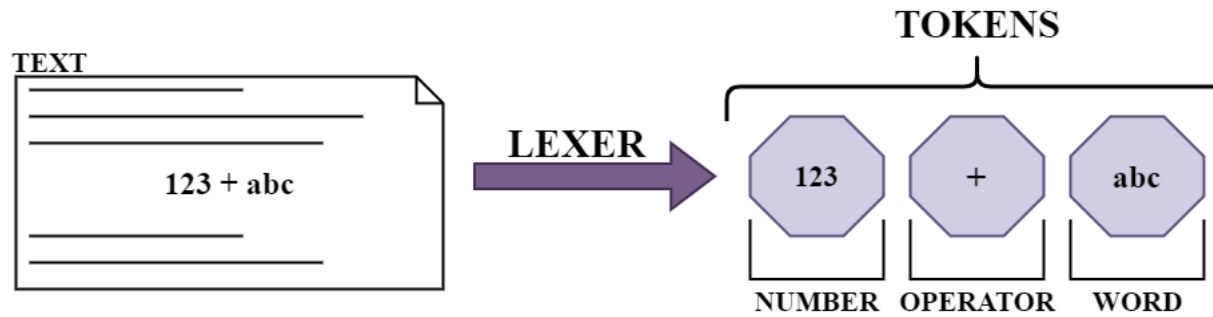


# Pièces de compilation





# Lexer



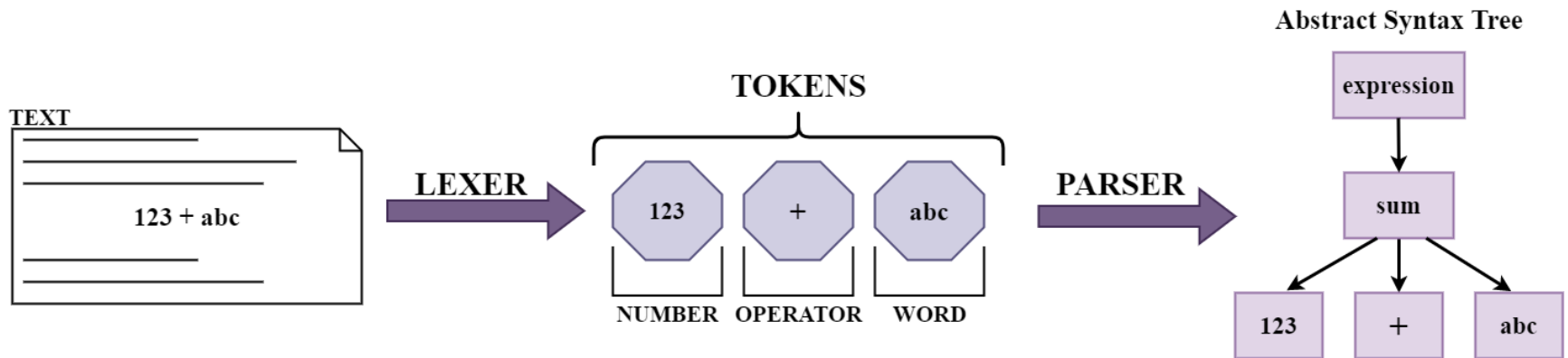
## Source

```
function s(a, b)
{
    return a+b;
}
```

## Jetons

```
FUNCTION: function
IDENTIFIER: s
LP: (
IDENTIFIER: a
COMMA: ,
IDENTIFIER: b
RP: )
LB: {
RETURN: return
IDENTIFIER: a
PLUS: +
IDENTIFIER: b
PV: ;
RB: }
```

# Analyseur



## Jetons

FUNCTION: function

IDENTIFIER: s

LP: (

IDENTIFIER: a

COMMA: ,

IDENTIFIER: b

RP: )

LB: {

RETURN: return

IDENTIFIER: a

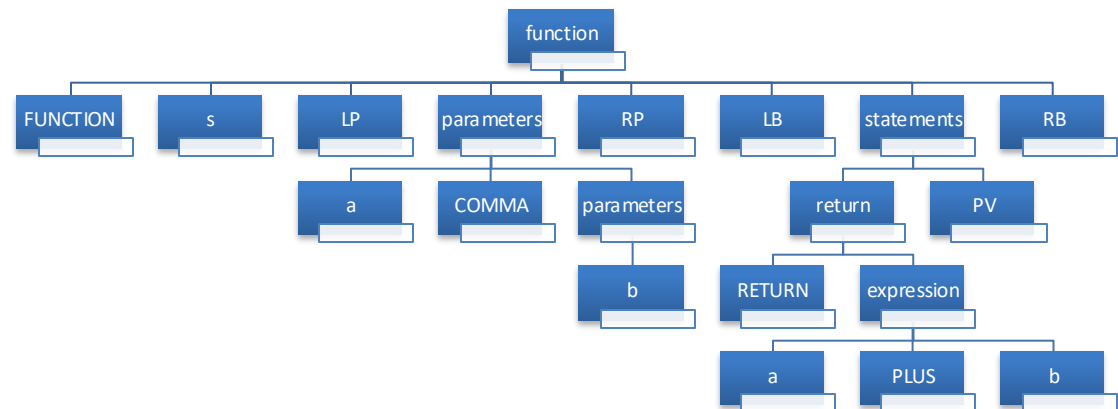
PLUS: +

IDENTIFIER: b

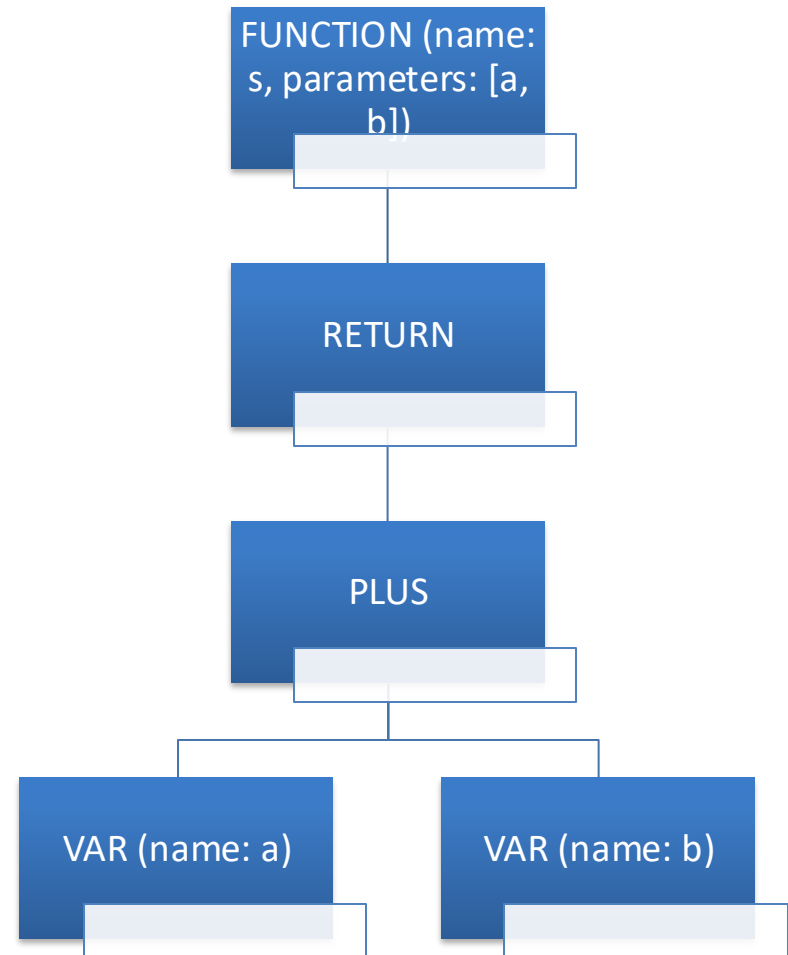
PV: ;

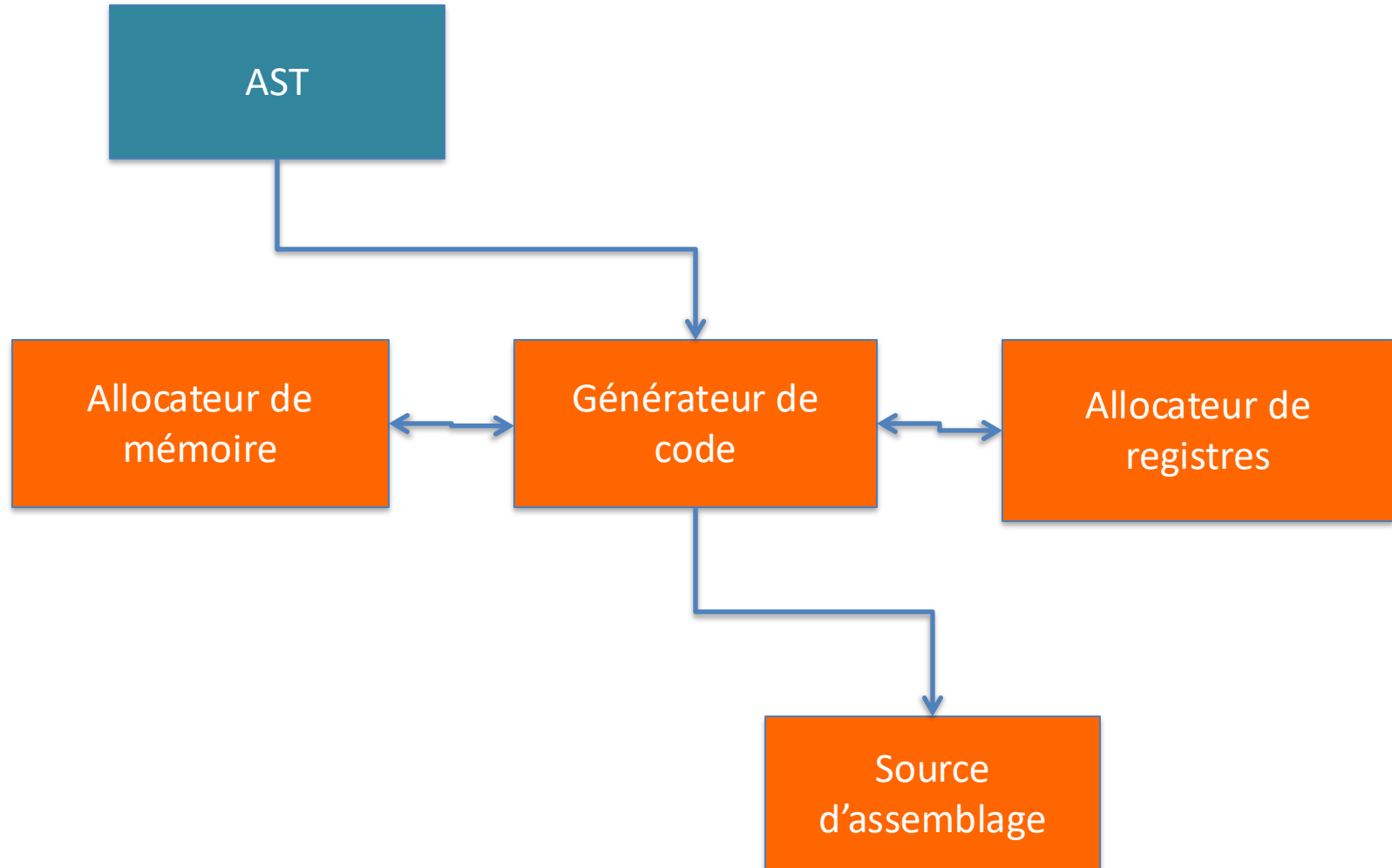
RB: }

## L'arbre d'analyse



```
function s(a, b)
{
    return a+b;
}
```







- Automates finités
- Expressions régulières
- Grammaires indépendantes du contexte
- Anayseur
- AST
- WebAssembly
- Représentation de la structure des données
- Génération de code

# Questions

---

