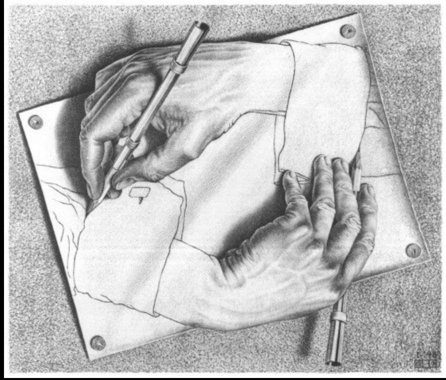
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE


Drawing Hands
M.C. Escher, 1948

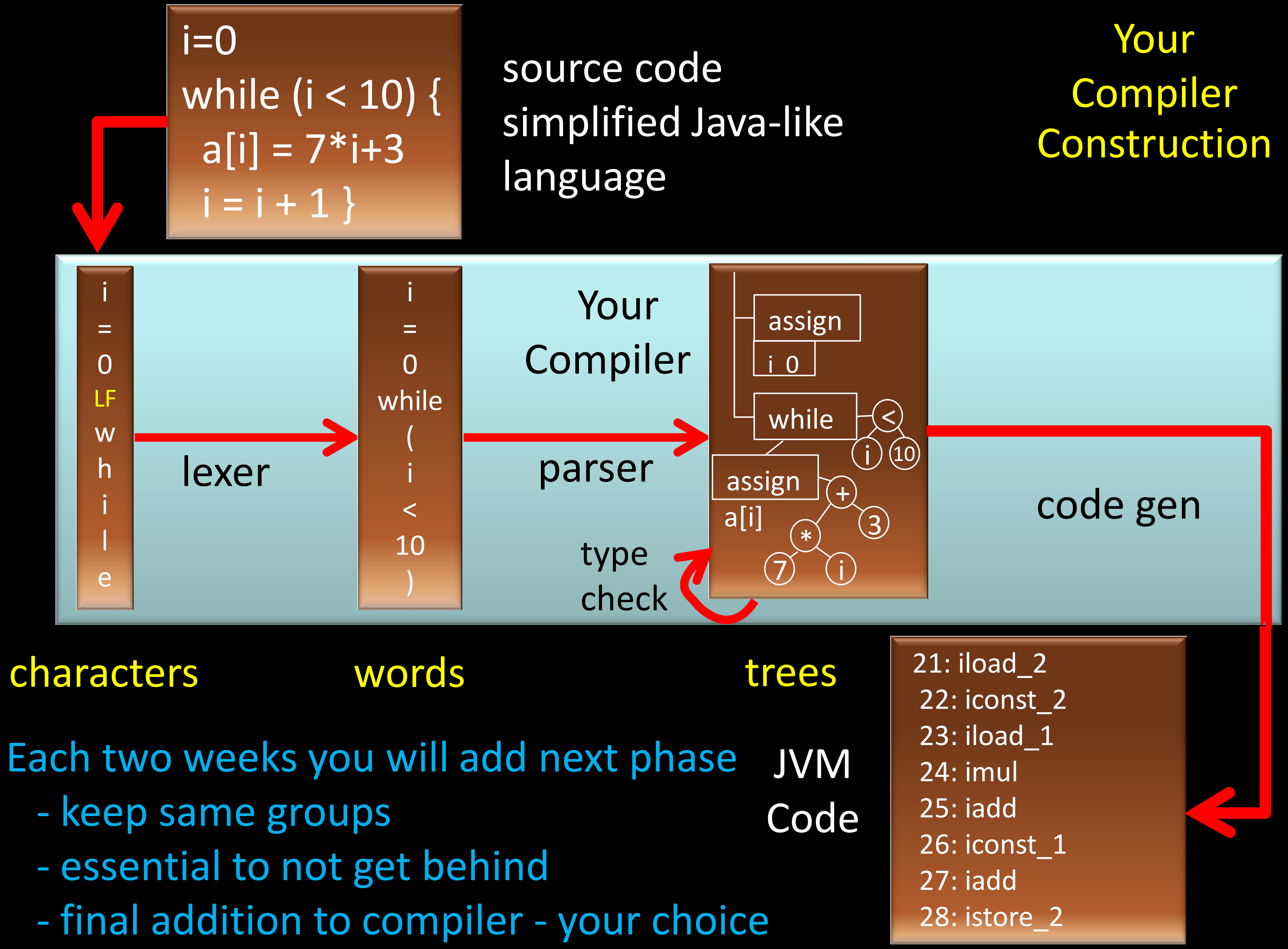
http://lara.epfl.ch

Compiler Construction **2010**, Lecture 2

Staff:

- Viktor Kuncak – Lectupes
- Hossein Hojjat – *Exercises*
- Philippe Suter – `{labs}`
- Étienne Kneuss, Ali Sinan Köksal – assistants
- Danielle Chamberlain – secretary

Your Compiler Construction



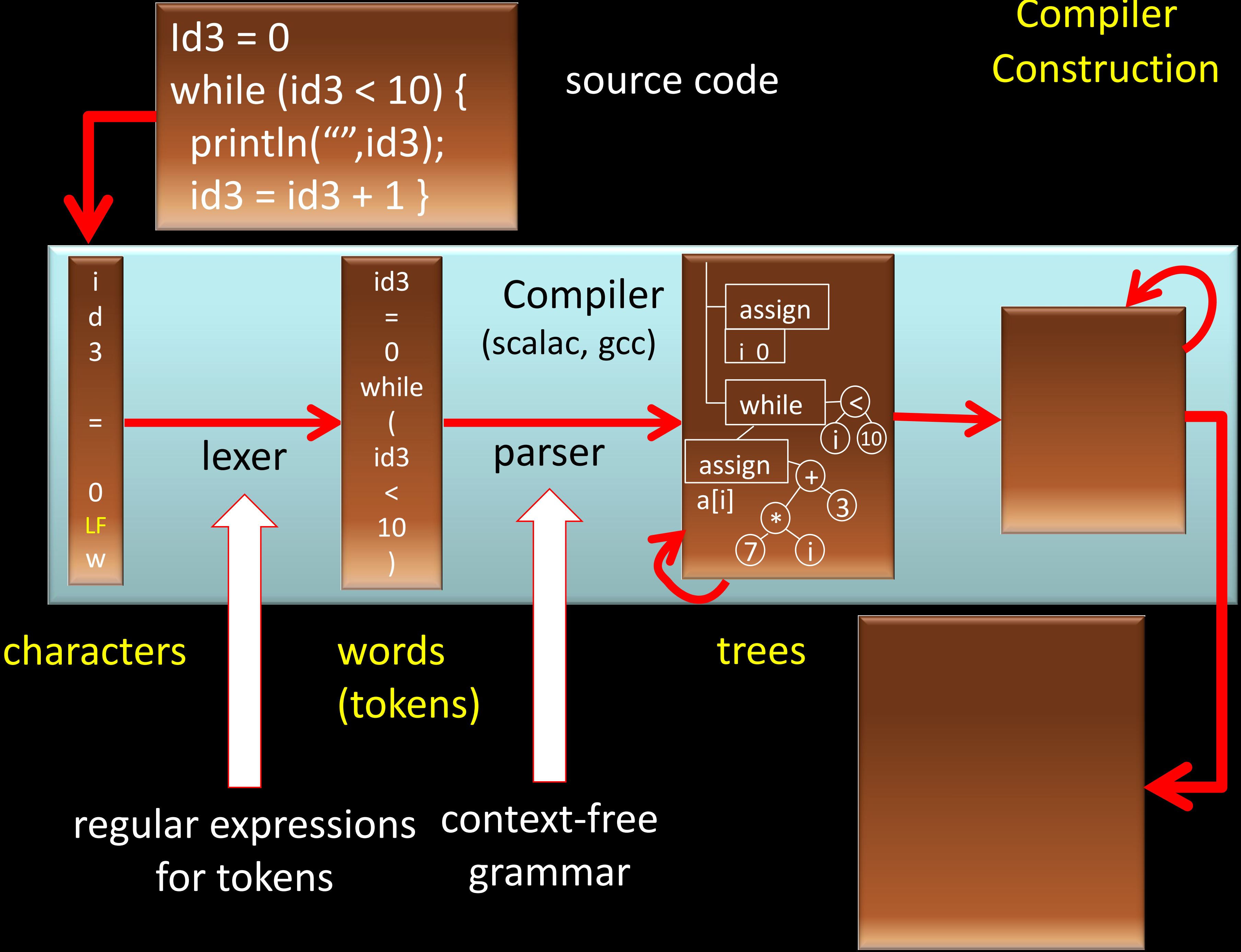
```
i=0
while (i < 10) {
  a[i] = 7*i+3
  i = i + 1 }
```

```
21: iload_2
 22: iconst_2
 23: iload_1
 24: imul
 25: iadd
 26: iconst_1
 27: iadd
 28: istore_2
```

Each two weeks you will add next phase

- keep same groups
- essential to not get behind
- final addition to compiler - your choice

Compiler Construction



Today

- Review
- Lexical analysis
- Idea of top-down parsing

Constructing Deterministic Automaton

- Automaton that accepts both binary and decimal numbers, where for binary numbers we use letter **o** instead of digit **0**

$$(\mathbf{o}|1)^* \mid (0|1|2|\ldots|9)^*$$

More Examples

- Find automaton or regular expression for:
  - as many digits before as after decimal point?
  - Sequence of open and closed parantheses of even length?
  - Sequence of balanced parentheses

    ```
    ( ( () )  ())      - balanced
    ( ) ) ( ( )        - not balanced
    ```
  - Comment as a sequence of space,LF,TAB, and comments from // until LF
  - Nested comments like /* ... /* */ ... */

Automaton that Claims to Recognize $\{ a^n b^n \mid n >= 0 \}$

We can make it deterministic

Let the result have K states

Feed it a, aa, aaa, ….

  consider the states it ends up in

More Examples

- Find automaton or regular expression for:
  - as many digits before as after decimal point?
  - Sequence of open and closed parantheses of even length?
  - Sequence of balanced parentheses

    ```
    ( ( () )  ())      - balanced
    ( ) ) ( ( )        - not balanced
    ```
  - Comment as a sequence of space,LF,TAB, and comments from // until LF
  - Nested comments like `/*  ... /*   */  ... */`

Limitations of Regular Languages

- Every automaton can be made deterministic
  - How?

$\delta(q, w)$ - state after reading w
$\in F$ iff w accepted

- Automaton has finite memory, cannot count
- Deterministic automaton from a given state behaves always the same
- If a string is too long, deterministic automaton will repeat its behavior
  - say A accepted $a^n b^n$ for all n, and has K states

$\delta(q_0, a^i) = \delta(q_0, a^{i+\lambda t})$ $\lambda, i \leq K$
$\delta(q_0, a^{K+1} b^{K+1}) = \delta(q_0, a^{K+1-\lambda} b^{K+1})$

Context-Free Grammars

- Σ - terminals
- Symbols with recursive defs - nonterminals
- Rules are of form
  N ::= v
  v is sequence of terminals and non-terminals
- Derivation starts from a starting symbol
- Replaces non-terminals with
  – terminals and
  – non-terminals

Balanced Parentheses Grammar

- Sequence of balanced parentheses

  ((()) ()) - balanced

  ( )) (( ) - not balanced

Recall While Syntax

```
program ::= statmt*
statmt ::= println( stringConst , ident )
           | ident = expr
           | if ( expr ) statmt (else statmt)?
           | while ( expr ) statmt
           | { statmt* }
expr ::= intLiteral | ident
         | expr (&& | < | == | + | - | * | / | % ) expr
         | ! expr | - expr
```

Eliminating Additional Notation

- Grouping alternatives
- Parenthesis notation

  expr (&& | < | == | + | - | * | / | %) expr
- Kleene star within grammars

  { statmt* }
- Optional parts

  if (expr) statmt (else statmt)$^?$