

Compiler Construction



Course Web Site:

<http://www.auc.auc.dk/~akbar/2011/complierconst-2011.html>

Sources details, schedule and all necessary information is provided here.

3

Dr. D. M. Akbar Hussain
Department of Electronic Systems

What is to be achieved



Basic principles of compiler construction and tools so that one can utilize these concepts may be to implement a compiler project or utilized the acquired knowledge for more general software engineering problems.

How:

- Discussion of the theory.
- Discussion on the various aspects of the theory.
- Examples.
- Solutions to selected Exercises with your participation.

Un-expected:

- **Modification (Normally additional stuff) of the text provided on the course web page.**

4

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Introduction

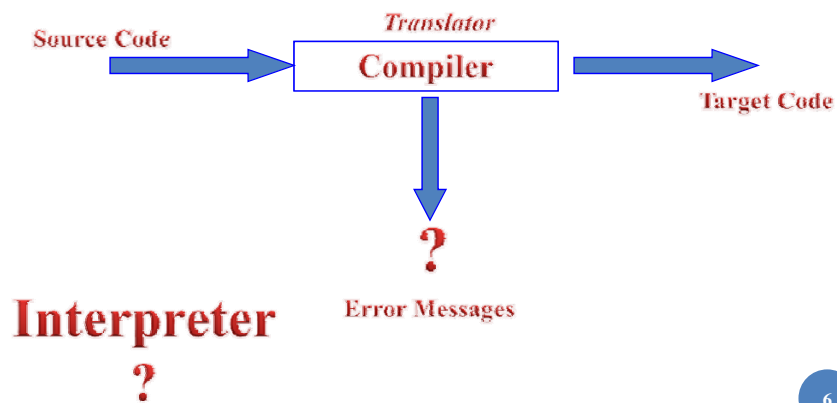


Compiler is a tool: which translate notations from one system to another, usually from source code (high level code) to machine code (object code, target code, low level code).

5

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Compiler



6

Dr. D. M. Akbar Hussain
Department of Electronic Systems

What is Involved




- **Programming Languages**
- **Formal Languages**
- **Regular Expressions & Automata Theory**
- **Applications**

7

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Programming Languages



- **We use natural languages to communicate**
- **We use programming languages to speak with computers**

8

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Formal Languages

```
graph TD; Grammar -- Define --> Language; Automaton -- Accept --> Language; Grammar -- Generate --> Automaton;
```

Regular Expressions & Finite Automata

Applications

- Editors
- Word-processors

9


Dr. D. M. Akbar Hussain
Department of Electronic Systems

Translation Flow

```
graph TD; SC[Source Code] --> S[Scanner]; S --> P[Parser]; P --> SA[Semantic Analyzer]; SA --> SCO[Source Code Optimizer]; SCO --> CG[Code Generation]; CG --> CO[Code Optimizer]; CO --> TC[Target Code]; LT[Literal Table] --- S; ST[Symbol Table] --- S; EH[Error Handler] --- S;
```

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Components of a Compiler




- Lexical Analysis
- Syntax Analysis
- Semantic Analysis

- Intermediate Code Generation
- Code Optimization
- Code Generation

11

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Lexical Analysis (LA)



Maradona kicks the ball

Token Generation:

- Maradona
- kicks
- the
- ball

Who performs this ?

12

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Lexical Analysis



X = Y + 30

Token Generation:

- **X** **id₁**
- **=** **operator**
- **Y** **id₂**
- **+** **operator**
- **30** **literal/constant**

What other functions Scanner can perform ?

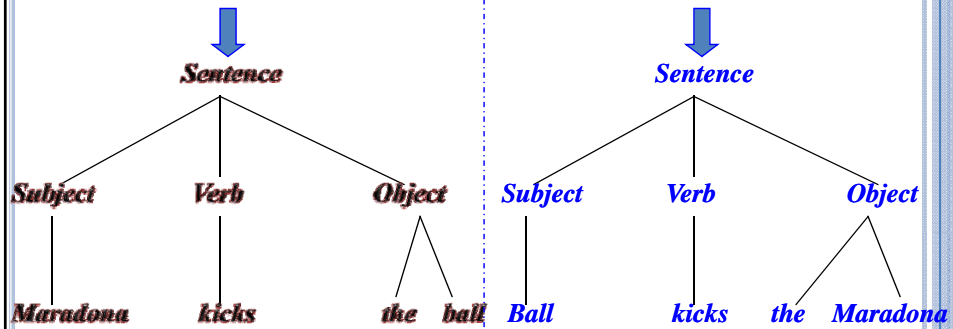
Syntax Analysis (SA)




Structure of the program is determined by SA. Some thing similar to grammatical analysis.

Maradona kicks the ball

Ball kicks the Maradona



Syntax Analysis (SA)



X = Y + 30

(Syntax Tree)


Who performs that ?

```
graph TD; E[Expression] --> AE[Assign Expression]; AE --> E1[Expression]; AE --> EQ[=]; AE --> AE2[Additive Expression]; E1 --> X[X]; AE2 --> P[+]; P --> E2[Expression]; P --> N[Number]; E2 --> Y[Y]; N --> 30[30];
```

Dr. D. M. Akbar Hussain
Department of Electronic Systems

15

Syntax Analysis (SA)



Some time syntax tree is also called as Abstract Syntax tree and could be a "trimmed" version of the parse tree with only essential information:

For Example: **a[index] = 4 + 2**

```
graph TD; AE[assign-expression] --> SE[subscript-expression]; AE --> A2[additive-expression]; SE --> I1[identifier]; SE --> I2[identifier]; I1 --> a[a]; I2 --> index[index]; A2 --> N1[number]; A2 --> N2[number]; N1 --> 4[4]; N2 --> 2[2];
```

Dr. D. M. Akbar Hussain
Department of Electronic Systems

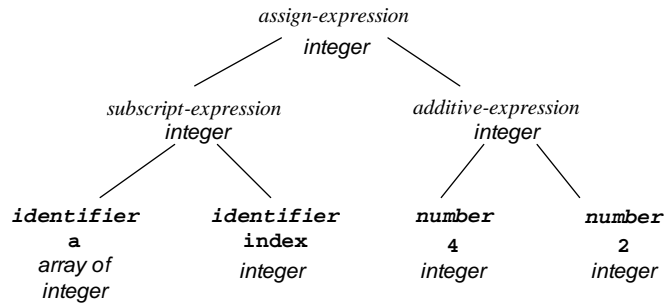
16

Semantic Analyzer



This attaches meaning to tokens; For example to the same expression

a[index] = 4 + 2



17

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Intermediate Code Generation



Same example: X = Y + 30

Temp1 = 30

Temp2 = Y

Temp3 = Temp2 + Temp1

X = Temp3

Three Address Code

P-Code

18

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Code Optimization



Same example: $X = Y + 30$

Temp1 = 30

Temp2 = Y

X = Temp2 + Temp1

Propagation ?

Constant folding ?

Common Sub-expression Elimination ?

Strength of Operation ?

19

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Propagation



```
int x = 10;  
int y = 7 - (x/2);  
return y * (100/x + 200);
```

```
int x = 10;  
int y = 7 - (10/2);  
return y * (100/10 + 200);
```

20

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Constant Folding



```
y = 10;  
for (i=0; i<10; i++)  
    x = (y + 2) + i;
```

```
y = 10;  
for (i=0; i<10; i++)  
    x = 12 + i;
```

21

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Common Sub-expression Elimination



```
A = x + b + c;  
B = 1 + y + b + c;  
C = b + c + z;
```

```
X = b + c;  
A = x + X;  
B = 1 + y + X;  
C = X + z;
```

22

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Strength Reduction



$2 * x \gg x + x$

$x * x \gg \text{shift left } 1$

Mathematical Identities

$a * b + a * c \gg a * (b + c)$

23

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Code Generation



Same example: $X = Y + 30$

Movei Y, r1

Addi 30, r1

Storei r1, X

24

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Algorithmic Tools



- **Token:**
 - Using Regular Expressions.
- **Scanner:**
 - Implementation of finite state machine to recognize tokens.
- **Parser:**
 - An Automaton (i.e. uses a stack), based on grammar rules in a standard format (BNF -- Backus Naur Form).
- **Semantic Analyzer and Code Generator:**
 - Recursive evaluators based on semantic rules for attributes (properties of language constructs).

25

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Portability Consideration



- **Front End:** Scanner, Parser, Semantic Analyzer and source code optimizer depend primarily on source language.
- **Back End:** Code generator and target code optimizer depend primarily on target language (machine architecture).
- Passes ?

26

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Data structures



- **Syntax tree: Kind of a link list structure**
- **Literal table: "Hello, world!", 3.141592653589793, etc.**
- **Symbol table: Names for variables, functions, classes, typedefs, constants.**

27

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Error handling



- **One of the difficult part of a compiler to design.**
- **Must handle a wide range of errors**
- **Must handle multiple errors.**
- **Must not get stuck.**
- **Must not get into an infinite loop.**

28

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Kinds of Errors



```
if (x == 0) y += z + r; }
```

Syntax:

```
int x = "Hello, world!";
```

Semantic:

```
int x = 2;
```

...

```
double y = 3.14159 / (x - 2);
```

Runtime:

29

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Error Handling Requirements




- A compiler must handle syntax and semantic errors, but not runtime errors (whether a runtime error will occur is a million dollar question).
- Sometimes a compiler is required to generate code to catch runtime errors and handle them in some graceful way (either with or without exception handling). This too is often difficult.

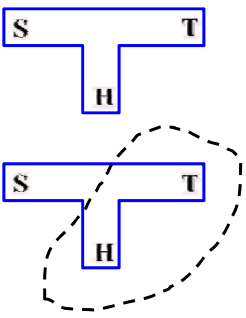
30

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Porting & Bootstrapping



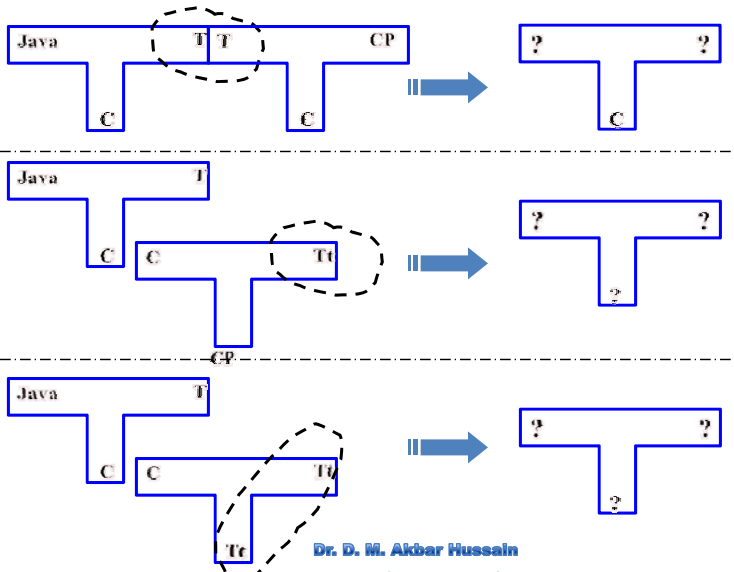

- o **Tool Chain:**
- o **T-Diagram:**



31

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Porting & Bootstrapping



32

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Rules

The diagram shows two stages of block placement. The top stage shows a horizontal row of blocks labeled CH, BC, CH, and BC. Below the first and third blocks are vertical lines leading to blocks labeled C. A dashed box encloses the two BC blocks, with a diagonal cross through it, indicating a rule or constraint. The bottom stage shows a more complex arrangement of blocks labeled C, BC, C, C, BC, and Adg. A dashed line traces a path through the blocks, and a blue arrow points from the left side to the right side, indicating a transformation or rule application. The Aalborg University Esbjerg logo is in the top right corner.

33

Dr. D. M. Akbar Hussain
Department of Electronic Systems

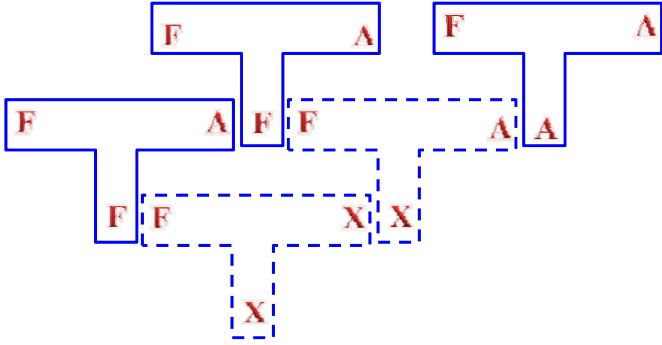
Steps of Bootstrapping

The diagram illustrates the steps of bootstrapping using blocks labeled F and X. A large dashed blue line encloses the entire structure. A dashed red line traces a path through the blocks, starting from the bottom and moving upwards. The blocks are arranged in a hierarchical structure, with F and X blocks connected by vertical and horizontal lines. The Aalborg University Esbjerg logo is in the top right corner.

34

Dr. D. M. Akbar Hussain
Department of Electronic Systems

Another Example



The diagram illustrates a parse tree for the string "FAFAFA". The root node branches into two children, "F" and "A". The "F" child branches into "F" and "A", while the "A" child branches into "F" and "A". This structure repeats, leading to a tree with a maximum depth of 3. The nodes are labeled with "F" and "A" in red. A dashed blue line outlines a path of nodes: the root's "F" child, its "A" child, the "F" child of that "A", and the "A" child of that "F". This path ends with a leaf node labeled "X". Another dashed blue line outlines a path: the root's "A" child, its "F" child, the "A" child of that "F", and the "F" child of that "A". This path ends with a leaf node labeled "X". The leaf nodes "X" indicate that the partial derivations along these paths are not complete or are invalid. The text "Another Example" is written in blue at the top left. The Alborg University Esbjerg logo is in the top right. The number "35" is in a blue circle at the bottom right. The name "Dr. D. M. Akbar Hussain" and "Department of Electronic Systems" are at the bottom center.

ALBORG UNIVERSITET
ESBJERG

35

Dr. D. M. Akbar Hussain
Department of Electronic Systems