Formal Language and Compiler Instructions
Design

E 75% 25%

C + S + L (+1 point from extra work)

In pandemia Teaching

Lab 1,2,3,4 compulsory (all 4 delivered in order to enter exam)
Lab 5 - optional

1) No copying!
2) Deadlines!
↳ maximal delay 2 more weeks (-1 point/week)

+1 point to final mark

Seminaries - ex and problems + small example for lab
- earn points / loose points
- homeworks (not compulsory)

C1

FLCD

Aho A.V., Sethi, Ullman - Compiler Principles, Techniques and Tools, Addison-Wesley, 1986

* Aho AV, Sethi - The Theory of Parsing, 1972 + 1973

Appel AW - Modern Compiler Implementation in C, Cambridge Univ Press, 1998

D. Grune, Bal, Jacobs, Langendoen - Modern Compiler Design, John-Wiley, 2002

L. Serbănati - Limbaje de Programare și Compilatoare, Ed Acad, 1987

S. Motogna - Metode de Proiectare a Compilatoarelor, Ed Albastra, 2006

G. Moldovan, M Lupea, V. Cioban - Limbaje formale, Culegere de Probleme source program → Compiler → object program

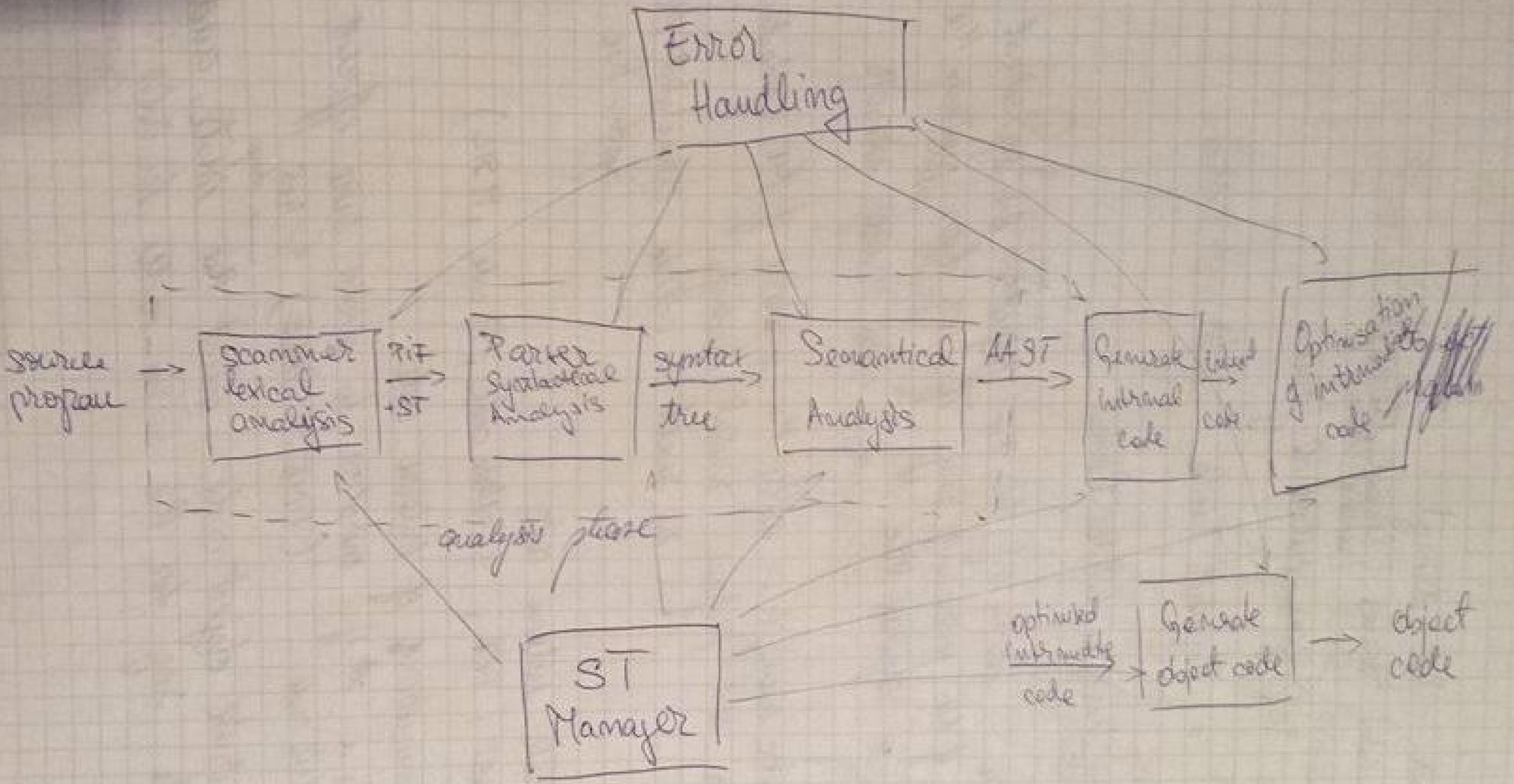
interpretor – line by line

dynamic compiler

source program → analysis → interme code → synthesis → object code

1st compiler 1957
FORTRAN
Backus

1st interpreter 1962
LISP McCarthy


Error Handling
source program
scanner lexical analysis
PIF
+ST
Parser Syntactical Analysis
syntax tree
Semantical Analysis
AAST
Generate internal code
interm code
Optimisation of intermediate code
analysis phase
ST Manager
optimized intermediate code
Generate object code
object code

1. Scanner – text (string)
(divide into) tokens (lexical items)
(atomi lexicali)

PIF = program internal form
+
ST = Symbol Table

Mary has a little lamb.

2. Parser – verify if the PIF respects the syntactical rules of the programming lang. If YES then produces the result = syntax tree

else ERROR

3. Semantic analysis – verify the meaning of the syntactical constr. and perform several checkings.

– the result is called Annotated Abstract Syntax Tree

4. Generate intern. code
(AAST, PIF, 3-address code, P-code, MSIL)

5. Optimisations of interm. code
– Is a set of operation to obtain more efficient intermediate code (TIME + SPACE) k = 4+5 ↳ keep

6. Generate object code – select the instr. and allocate reg. for the intern. code in order to generate the code.

Symbol Table Manager – create / access / update operations for symbol table

Error Handling – go specific to detect errors for each phase

REMARK

Each phase is an equivalent transformation of its input into its output