Tutorial 11

Page No.
Date / /

PRN :- 2018BTECS00101

* Make a report on LANCE Re-targetable C compiler.

→

1] Introduction :

LANCE is a software platform for fast implementation of C compilers. It includes C frontend, a set of code optimization passes on intermediate representation levels, a C library for IR analysis, and manipulation, backend for assembly code generation. LANCE is mainly intended for C compilers development for embedded process.

2] Software Architecture :-

LANCE consist of C++ library and compiler tools. The C++ library provides an API for IR access and manipulation. In addition, it comprises numerous algorithms and data structures needed for building compiler tools, ranging from simple structures to advanced code analysis routines. All LANCE tools are built on C++ library. They operate on a common IR in the IR-C format in plug-and-play fashion. While LANCE already comprises a set of IR optimization tools, new IR transformations can be easily added

BTECS00101

etable

or fast
t includes
mization
ation levels
nd
mbly code
ed for C
dded proces

l compiler
n API for
addition,
ms and
ding compiler
tures
es
t library,
in the
y fashion
a set
o IR
ly added

anytime.

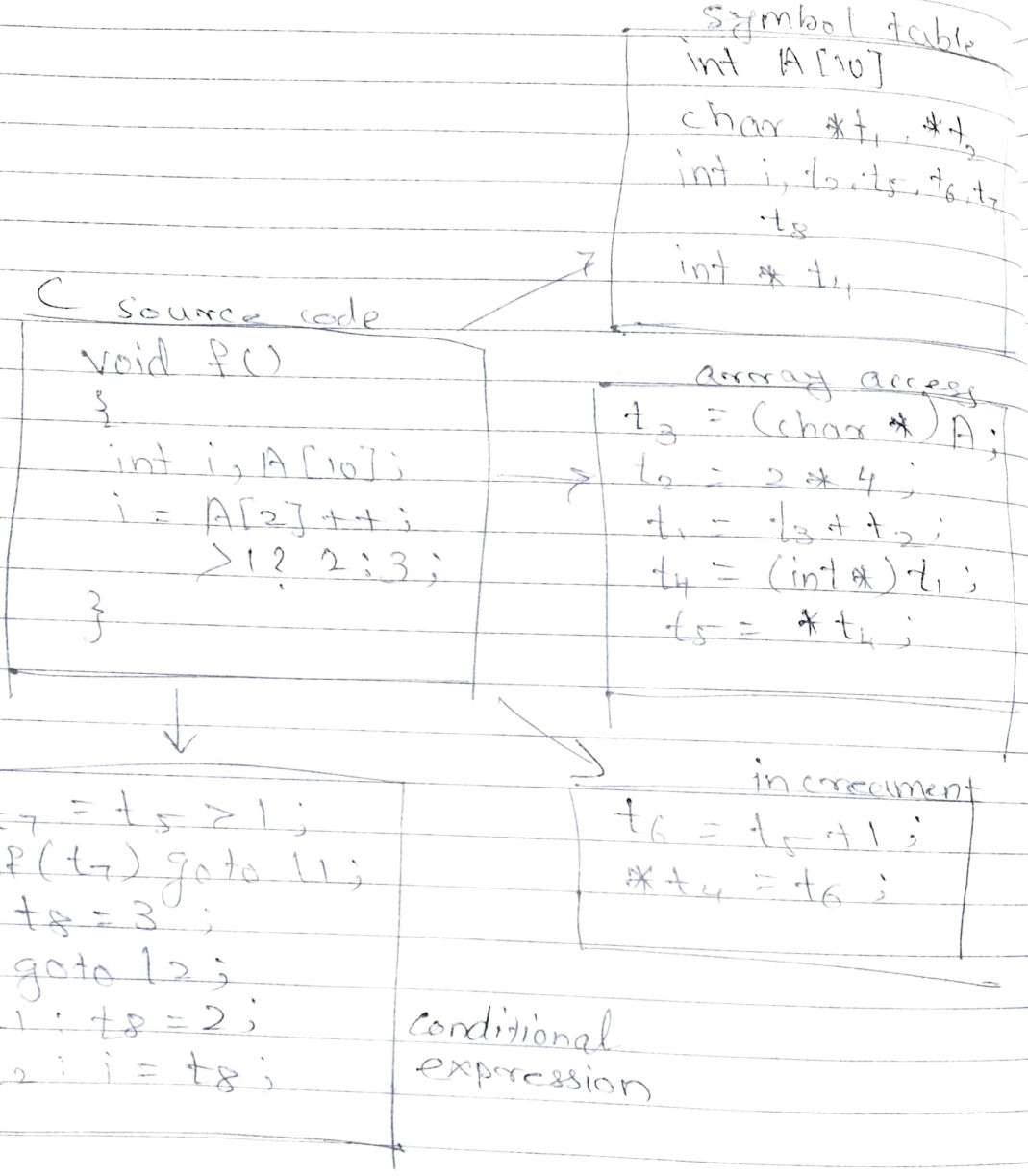
3] ANSI C frontend :-

LANCE comprises a frontend for full ANSI C. It requires a separate C preprocessor though and standard header files and libraries are not included. Normally these components are recieved from GNU C compiler.
LANCE's ANSI C frontend source code has been automatically generated by means of the attribute grammar compiling system OX. OX reads an attribute grammar and generates lex and source file.

4] Intermediate Representation (IR) :-

LANCE uses a 3 address code IR.
Complex expressions are decomposed into 3 address code by means of insertion of temporary variables. High level control flow statements are replaced with equivalent jump constructs close to assemble level.
The IR can be dumped into low level ANSI C syntax. This implies a clear & easy to learn IR semantics and enables executability of IR.

Page No.
Date



5] IR-level code optimization engine:

LANCE includes a number of both classical and advanced IR-level code optimization engines. They are implemented as separate tools and can be

6] Flow

DF

t1

t4

if(t

go

Page No.
Date / /

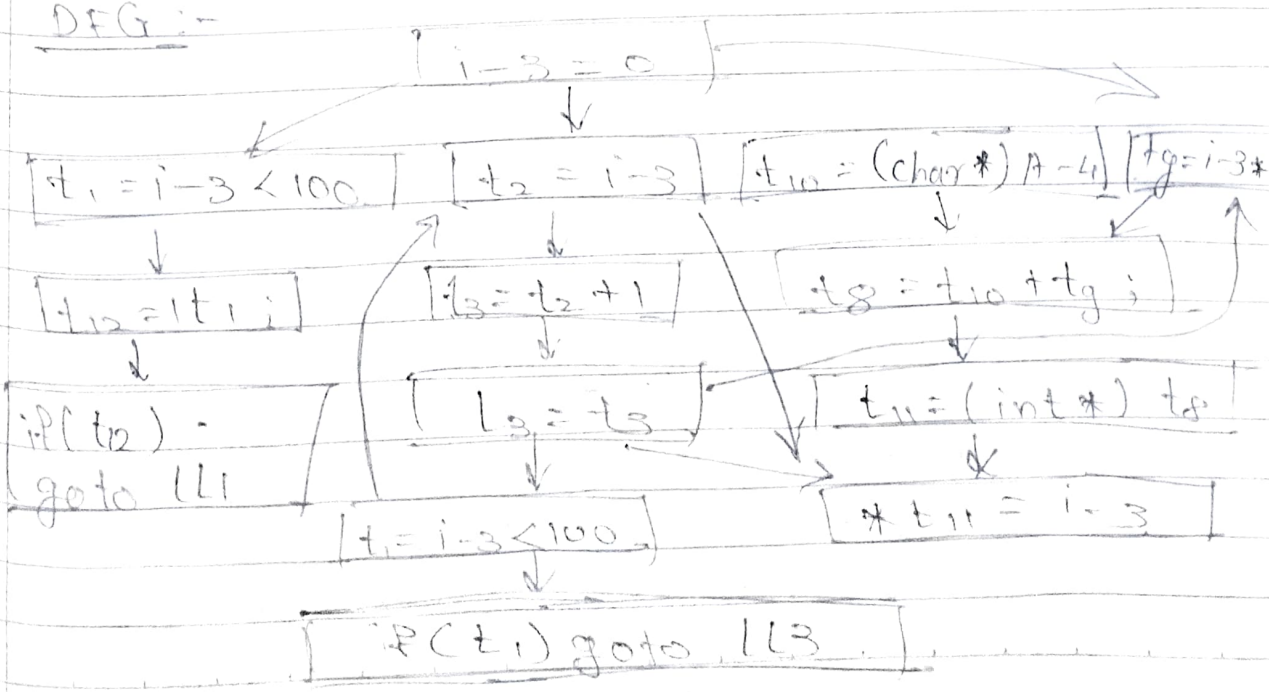
combined and iterated by predefined IR optimization script that can be modified or extended.

Currently, following optimization engines are available,

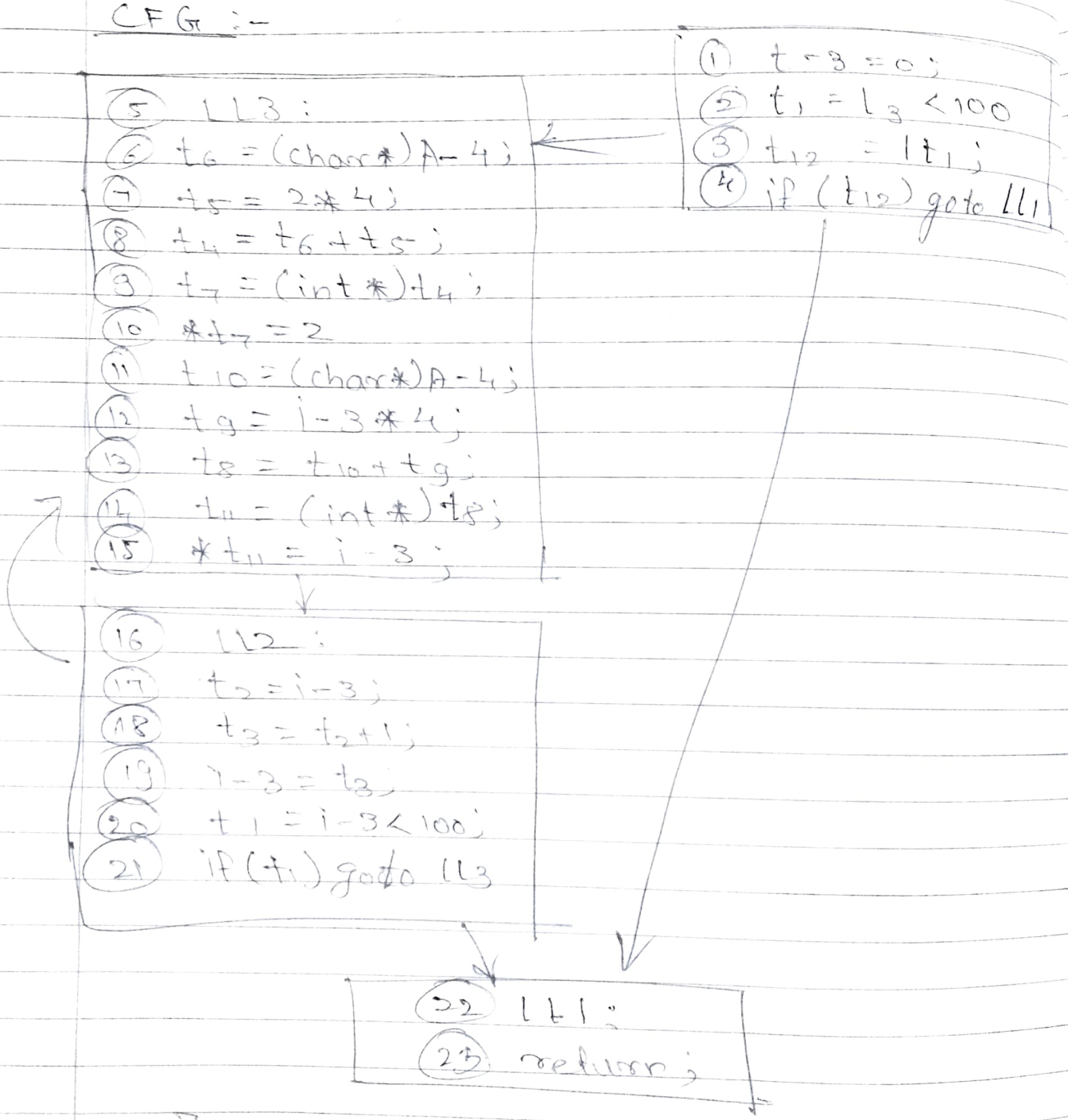
i) Constant folding
ii) copy propogation
iii) constant propogation
iv) dead code elimination.
v) Common subexpression elimination.
vi) SSA conversion
vii) Function inlining.
viii) induction variable optimization.
ix) jump optimization.
x) loop invariate code motion
xi) reassociation.

6] Flow graph (CFG, DFG) :-

DFG :-



CFG :-



7) Verification :-

LANCE permits plug and play integration of new IR transformation and optimize passes. Naturally, such passes needs

to be extensively verified. LANCE supports verification of any IR transformation by means of its executable IR format without need for compiler backend or instruction set simulator.

8] API for code instrumentation at IR level :-

LANCE includes an API for code instrumentation at IR level, which supports implimentation of advanced applicat code profiling tools. The API permits the automatic placements of user defined probes func calls betn 3 address code IR statements generated by C frontend.

9] Backend Interface :-

LANCE can convert its 3 address code IR into a tree data structure that is compatible to popular code generator tools like iburg and olive. This feature facilitates compiler retargeting as designer can directly start by writing target specific code selector specification.