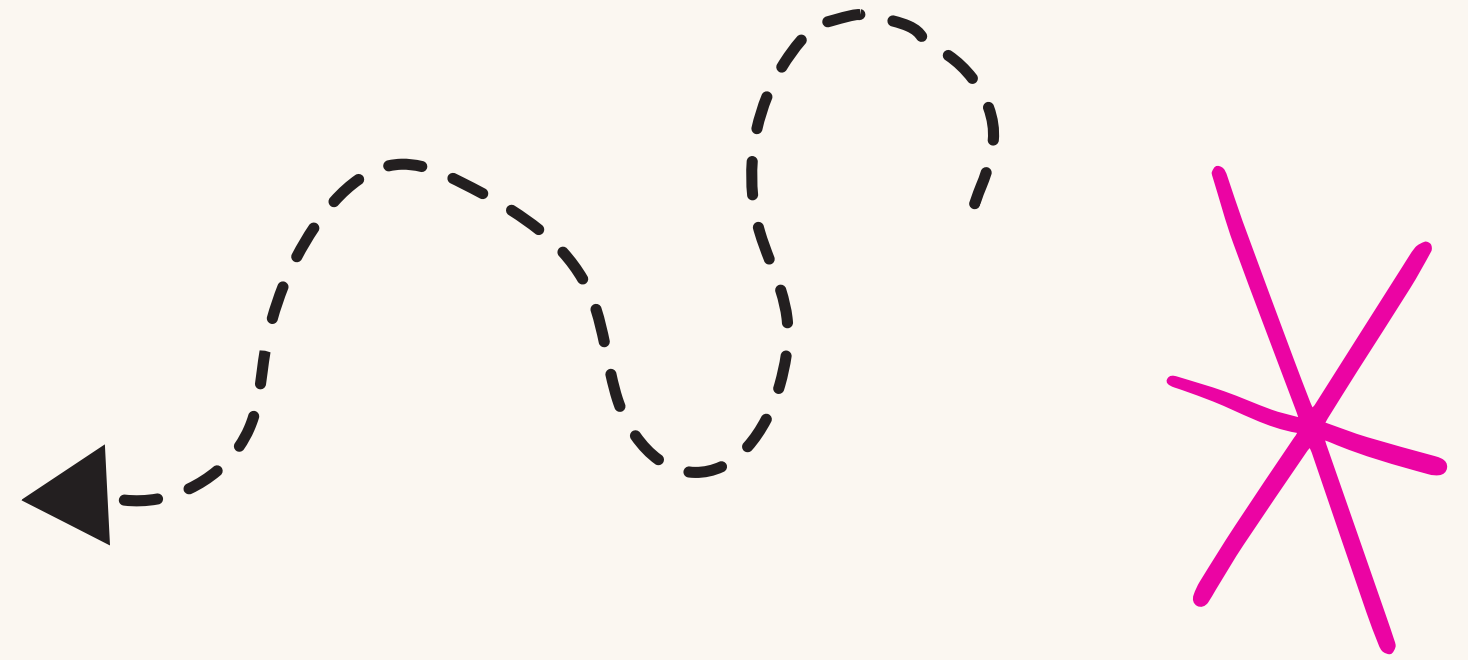


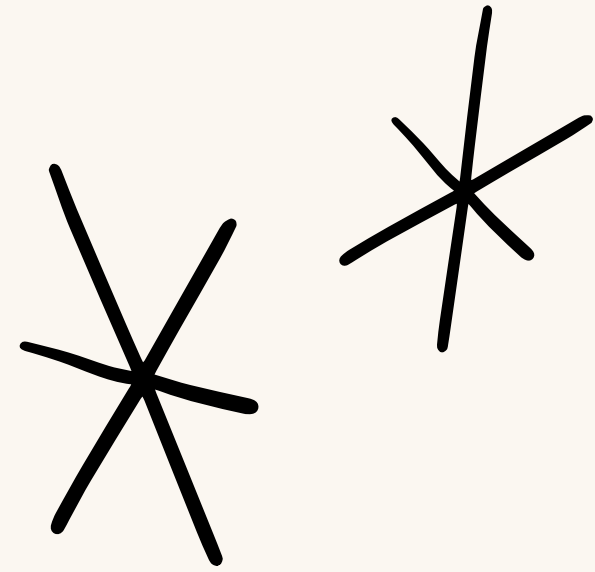
compiler PHASES





Agenda

1. Introduction
2. Types of compiler
3. compiler phases
4. cfg design
5. project implementation
6. conclusion

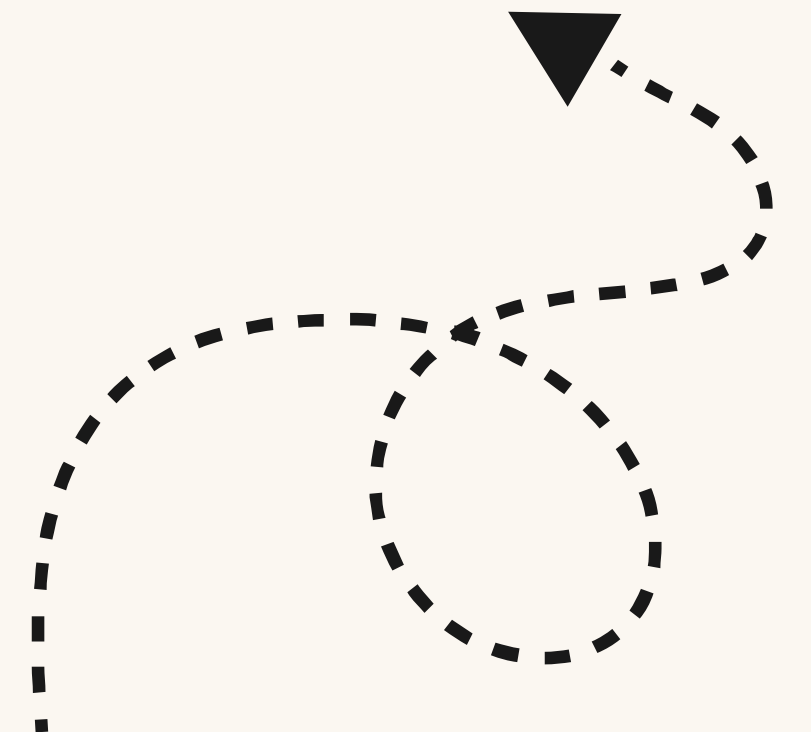


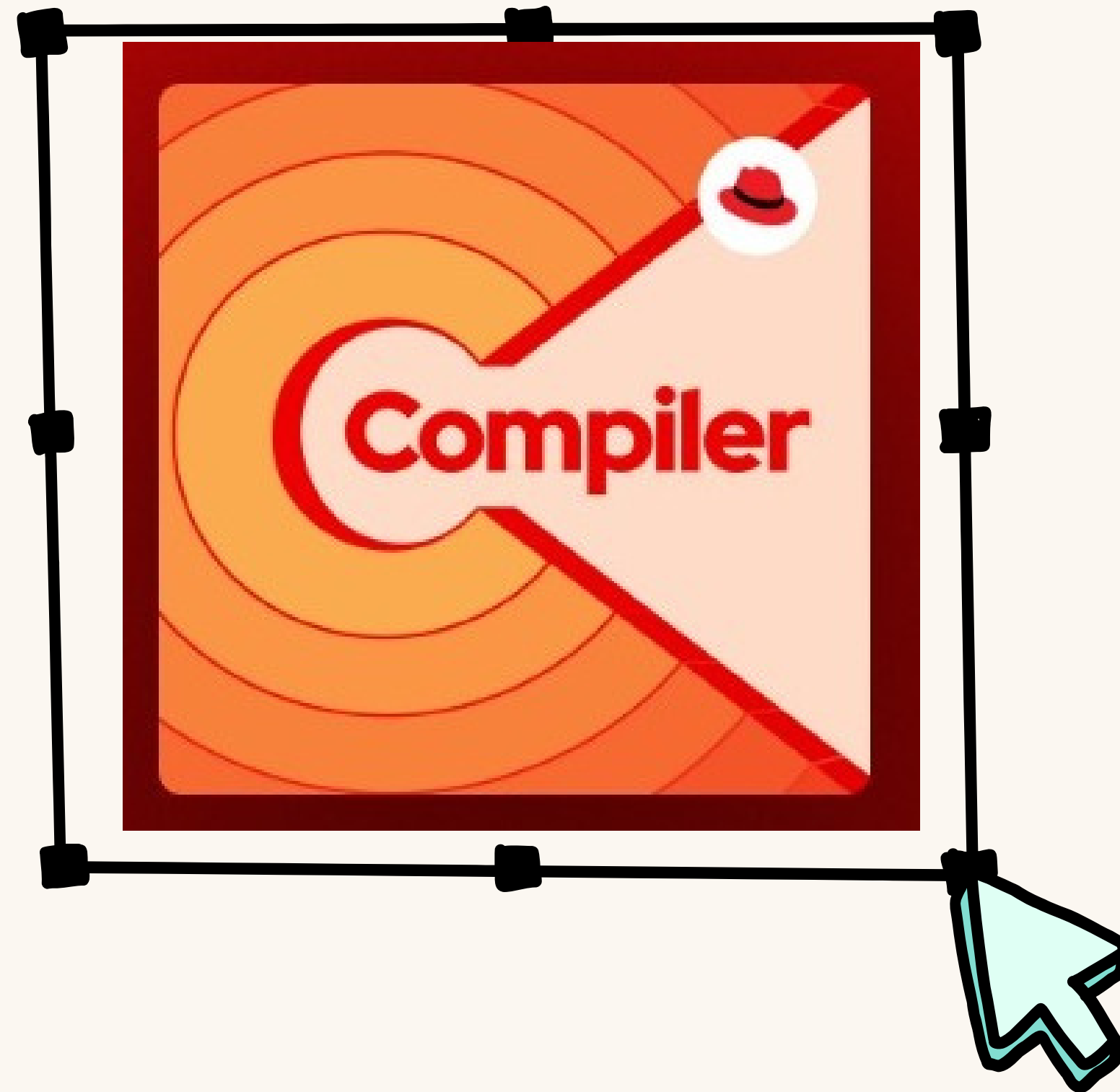
I AM



basant medhat makram

20213771





Introduction



It is special program that translates a programming language's source code into machine code, bytecode or another programming language. contains the sequence of various phases.



TYPES OF COMPILER

1

single
pass

2

Multi
pass

3

Just-
in-time

4

CROSS
COMPILER

5

AHEAD
OF TIME

Lexical analyzer

**intermediate
code generator**

syntax analyzer

**optimizer
code generator**

Semantic analyzer

code generator

COMPILER PHASES



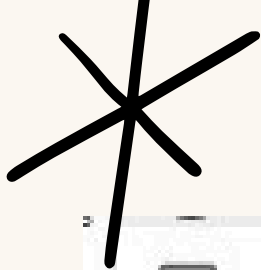
```
graph TD; C[COMPILER PHASES] -.-> LA[Lexical analyzer]; C -.-> SA[syntax analyzer]; C -.-> SEM[Semantic analyzer]; C -.-> ICG[intermediate code generator]; C -.-> OCG[optimizer code generator]; C -.-> CG[code generator];
```

The diagram illustrates the six phases of a compiler. The central text 'COMPILER PHASES' is written in a large, bold, black font. Surrounding this central text are six colored rounded rectangular boxes, each containing the name of a compiler phase. Dashed black arrows point from the central text to each of these boxes. The boxes are arranged in a circular pattern around the center. The colors of the boxes are: Lexical analyzer (light blue), syntax analyzer (orange), Semantic analyzer (purple), intermediate code generator (yellow), optimizer code generator (pink), and code generator (teal). There are also several pink brushstroke-like lines radiating from the central text area.

snapshots of running

Run of the code







Input:

Form: ☒ Math Form ☐ Source Form

Submit

Lexical Analysis

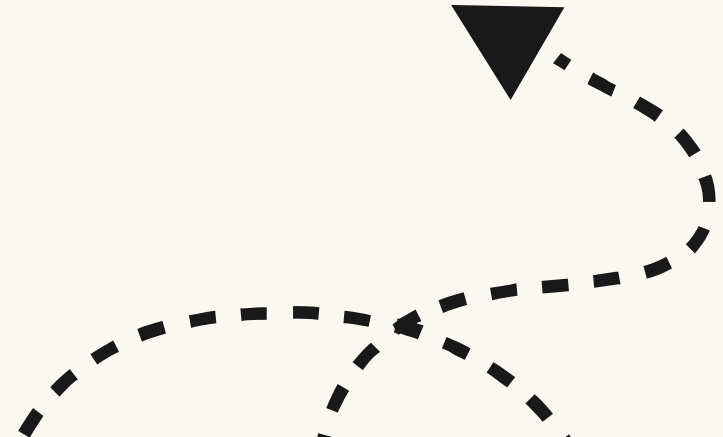
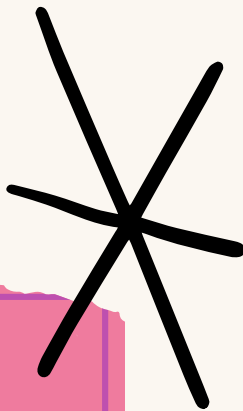
Syntax Tree

Semantic Tree

Intermediate Code

Optimized Code

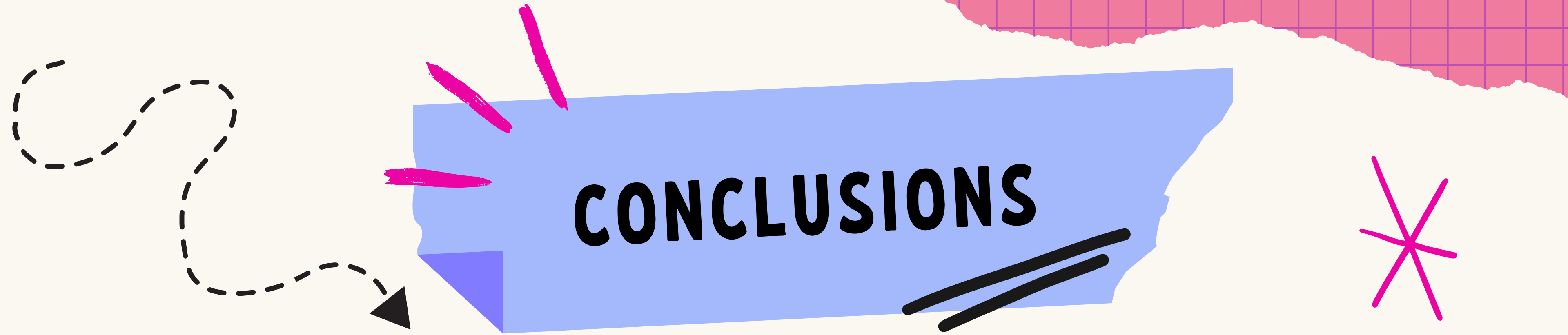
Machine Code



CFG DESIGN

PRODUCTION RULES

1. $\text{Exp} \rightarrow \text{E Add_op term}$
2. $\text{Exp} \rightarrow \text{term}$
4. $\text{Term} \rightarrow \text{Term Mulop Factor}$
5. $\text{Term} \rightarrow \text{Factor}$
5. $\text{Factor} \rightarrow (\text{Exp}) \mid \text{id} \mid \text{num}$
6. $\text{Mulop} \rightarrow *, /$
7. $\text{add_op} \rightarrow +, -$



CONCLUSIONS

At the end we know how compilers work by breaking down each phase, analyzing code to generating machine instructions.

In the future, we could expand the compiler to support more complex programming constructs and optimize performance further

“ The most dangerous phrase in the language is, 'We've always done it this way.'" By learning compiler design, we challenge and innovate the way code is translated and executed. “

Grace Hopper

THANK
YOU!

