ASSIGNMENT-I COMPILERS

(Itish Agarwal, 18CS30021)
14 Sep'2020

Q1. We have,

int a = 5, b = 6, c = 2, d;
d = b * c + a;

↓

Lexical Analysis

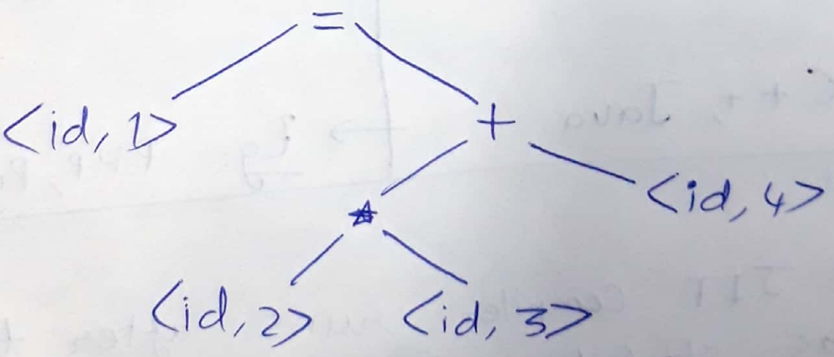
↓

<id, 1><=><id, 2><*><id, 3><+><id, 4>

↓

Syntax Analysis

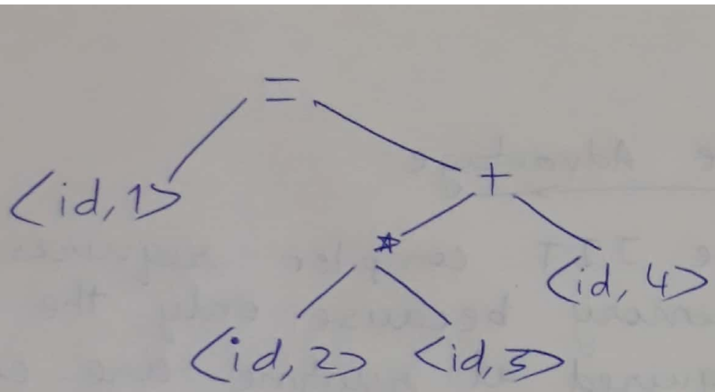
↓



↓

Semantic Analysis

↓



As all variables are of type int, we do not to typecast any

Intermediate code Generation

t1 = id2
t1 = t1 * id3
t1 = t1 + id4
id1 = t1

Code optimisation

t1 = id2 * id3
~~t1~~
id1 = t1 + id4

Q2.

COMPILER	INTERPRETER
→ It considers the completion of the program as input for converting to machine code.	→ It considers one statement in the program at a time as input for converting to machine code.
→ faster execution of control statements as compared to interpreter.	→ Slower execution of control statements as compared to the compiler.
→ Does not generate intermediate code. Hence, an interpreter is highly efficient in terms of its memory.	→ A compiler always generates an intermediate code. It will need further linking. Hence more memory is needed.
→ Eg: C++, Java	→ Eg: PHP, Python

Q3. A JIT Compiler runs after the program has started and compiles the code (usually bytecode or some kind of VM instructions) on the fly into a form that's usually faster, typically the host CPU's naive instruction set.

Eg: It is an essential part of JRE (Java Runtime Environment)

Performance Advantage:

The JIT compiler requires less memory because only the methods required at runtime are compiled into machine code. Page faults are reduced by using JIT compiler.