Name: Sumit Kumar Yadav
Roll No. 18CS30042

Assignment-1

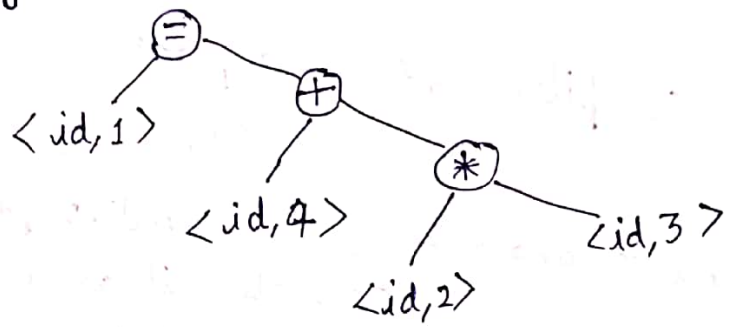
1. Discuss the phases of a four-pass compilation of the below C program:

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

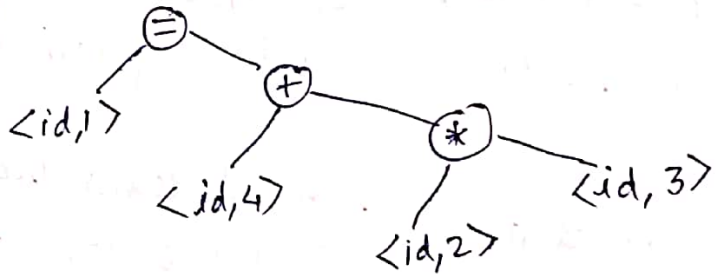
⇒ Lexical Analyzer:

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

Syntax Analyzer:



Semantic Analyzer:



Intermediate Code Generator:

```
t1 = id2
t2 = id3
t3 = t1 * t2
t4 = id4
t5 = t3 + t4
id1 = t5
```

Code Optimizer :-

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

Code Generator :

```
LDI   R2 , id2
MULI  R2 , R2 , id3
LDI   R3 , id3
MULI  R2 , R2 , R3
LDI   R1 , id4
ADDI  R1 , R1 , R2
STI   id1 , R1
```

2. Difference between Compiler and Interpreter :-

Compiler	Interpreter
→ Scans the entire program & translates it as a whole into machine code	→ Translate program one statement at a time
→ It converts the translations into systematic code.	→ It doesn't convert the instructions instead it directly works on source language
→ As it scans the code in one go, the errors are shown at the end together.	→ It scans code one line at a time, errors are shown line by line.
→ Main advantage of compilers is its execution time	→ Interpreter are slow in executing the object code.
→ It generates intermediate object code which further requires linking, hence requires more memory	→ No intermediate object code is generated, hence are memory efficient.
→ Programming language like C, C++, Java, C# use compilers.	→ Programming language like Python, Ruby, Javascript, Perl, MATLAB use interpreter

③ Just-In-time Compilation :-

Just-in-time compilation is a way of executing computer code that involves compilation during execution of a program at run time rather than before execution. Most often, this consists of source code or more commonly bytecode translation to machine code, which is then executed directly.

A System implementing a just-in-time compiler typically continuously analyses the code being executed and identifies parts of the code where the speedup gained from compilation or recompilation would outweigh the overhead of compiling that code.

- For example, in Java programming language and environment, a just-in-time compiler turns Java bytecode (a program that contains instructions that must be interpreted) into instructions that can be sent directly to the processor. It is an essential part of JRE ie Java runtime Environment, that is responsible for performance optimization of java based applications at run time.
- In terms of performance, JIT compilers interact with JVM at run time and compile suitable bytecode sequences into native machine code. This subsequently leads to performance gains in the execution time, unless the compiled methods are executed less frequently. Some of the optimizations performed by JIT compilers are data-analysis, reduction of memory accesses by register ~~allocation~~

allocation, translation from stack operations to register operations, elimination of common sub-expressions etc.

JIT causes a slight to noticeable delay in initial execution of an application, due to the time taken to load and compile the bytecode. sometimes this delay is called 'startup time delay' or 'warm-up' time

In general, the more optimization JIT performs the better the code it will generate, but the initial delay will also increase.