Principles of Programming Languages(CS F301)

BITS Pilani
Hyderabad Campus

Prof.R.Gururaj
CS&IS Dept.

Computer System

What a computer system includes

- ❑ *Processor(CPU)*
- ❑ *Memory*
- ❑ *Input- Key board, Mouse etc.*
- ❑ *Output- Monitor, Printer etc.*
- ❑ *Memory (sec)*
- ❑ *OS and*
- ❑ SW

Computers for Solving Problems

- ❑ Problem statement
- ❑ Algorithm
- ❑ Program
- ❑ Machine instructions
- ❑ Execution

Algorithm

Is a step-by-step procedure (to solve problems) generated to terminate such that each step is precisely stated and carried out by the computer.

Is a finite set of instructions which if followed accomplishes a particular task.

Low-level Language

Low-level Language is tied to the computer hardware (machine dependent).

Each computer (HW) will have one such low-level language we call such language as Assembly Language (in addition to its Machine Language).

High-level Language

High-level Languages are at a distance from the computer(machine independent)

High-level programming languages allow the specification of a problem solution in terms closer to those used by human beings.

These languages were designed to make programming far easier, less error-prone and to remove the programmer from having to know the details of the internal structure of a particular computer.

Advantages of High-level Programming Languages

- ❑ High-level languages allow us to use symbolic names for values.
- ❑ High-level languages provide expressiveness.
- ❑ High-level languages enhance readability.
- ❑ High-level language provide abstraction of the underlying HW.
- ❑ High-level languages safeguard against bugs.

Some important Programming Languages

BASIC, PROLOG, LISP

C,

C++,

Cobol,

FORTRAN,

Java,

Pascal,

Perl,

PHP,

Python,

Ruby, and Visual Basic.

Principles of Programming Languages (PPL)

Preliminaries (Ch.1 of T1)

Why to study Concepts of Programming languages ?

1. Increased Capacity to express ideas.
2. Improved background for choosing appropriate PL.
3. Increased ability to learn new languages.
4. Better understanding of the significance of implementation.
5. Better use of languages you already know.
6. Overall advancement of Computing.

Programming Domains

- *Scientific Applications*

  Arrays, Matrices, Loops, selections, large number of Floating point operations are the requirement.

  Fortran (***FOR****mula* ***TRAN****slation) 1950s; ALGOL 60*
- *Business Applications*

  Producing reports, precise way to specify and store decimal and character data.

  COBOL (**CO**mmon **B**usiness-**O**riented **L**anguage).
- *Artificial intelligence*

  LISP (LISt Processing) 1965.
- *System Programming*

  OS and other support tools of computer are called as System SW.

  UNIX is completely written in C (ISO 1999).
- *Web Software*

  PHP, Python, Java Script with HTML

Language evaluation Criteria

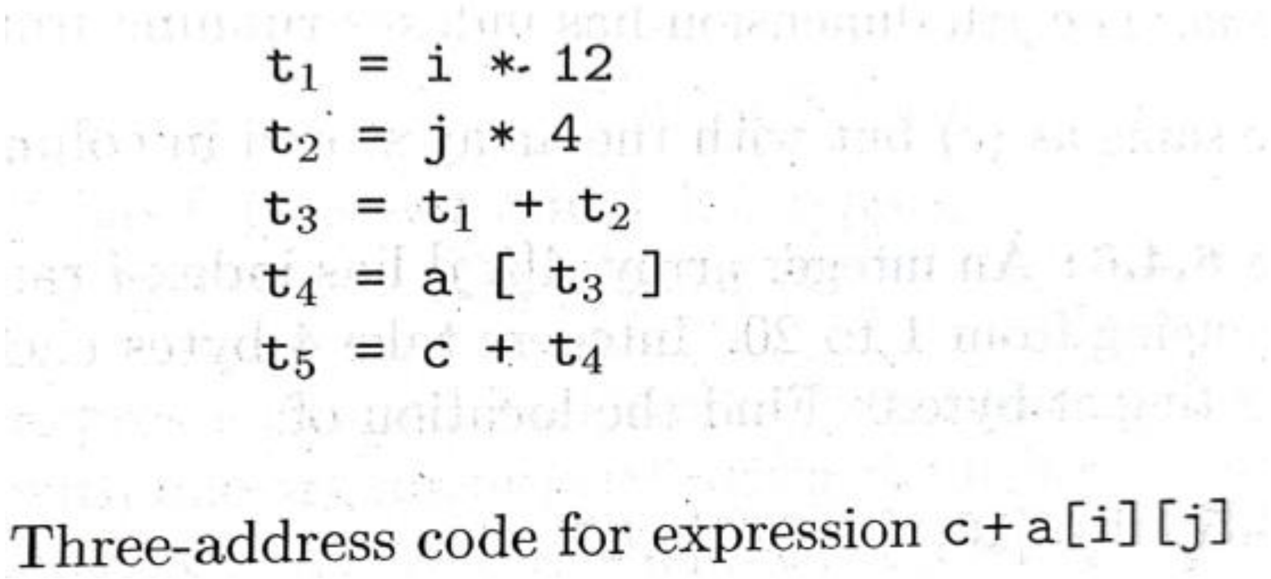
1. Readability
2. Writability
3. Reliability
4. Cost

1.Readability is affected by -

- *Overall simplicity*
- *Orthogonality*
- *Data types*
- *Syntax Design*

```
position = initial + rate * 60
```

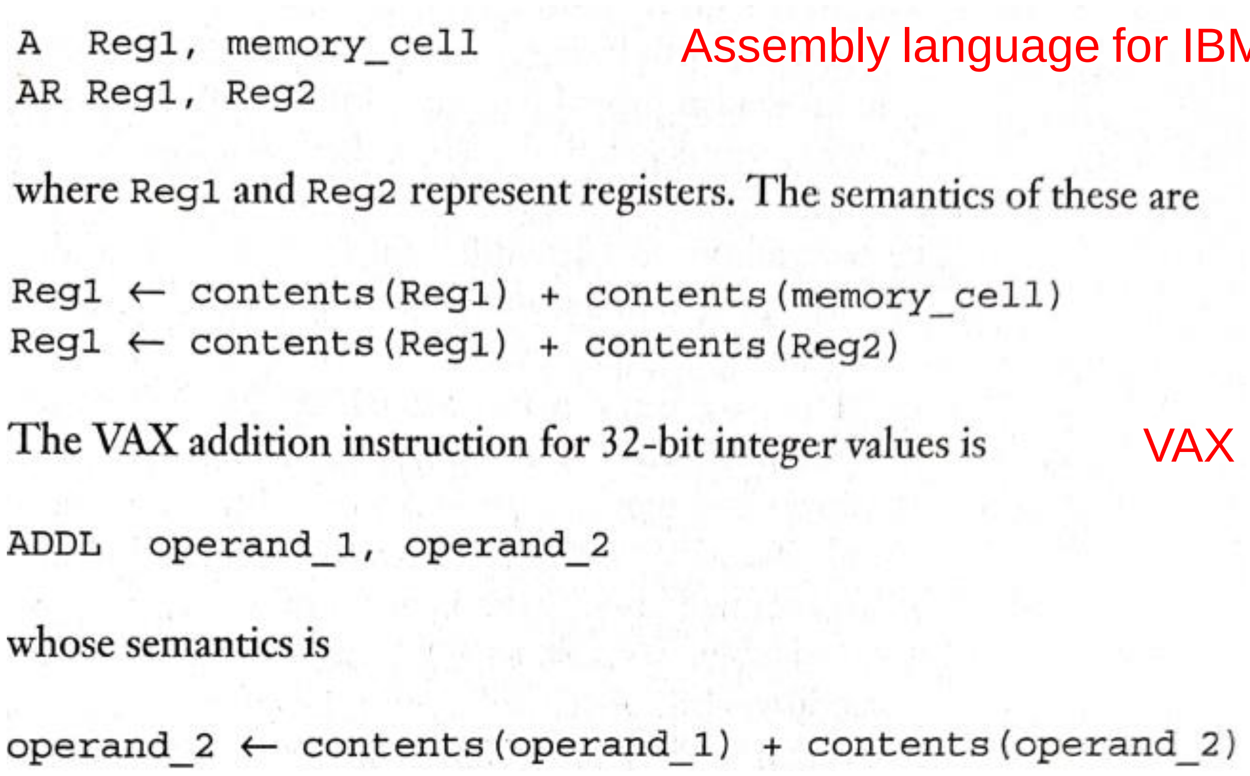
```
LDF  R2, id3
MULF R2, R2, #60.0
LDF  R1, id2
ADDF R1, R1, R2
STF  id1, R1
```



$$
\begin{aligned}
t_1 &= \texttt{i} * 12 \\
t_2 &= \texttt{j} * 4 \\
t_3 &= t_1 + t_2 \\
t_4 &= \texttt{a} [t_3] \\
t_5 &= \texttt{c} + t_4
\end{aligned}
$$

Three-address code for expression `c+a[i][j]`

Example for Orthogonality



```
A   Reg1, memory_cell
AR  Reg1, Reg2
```

where Reg1 and Reg2 represent registers. The semantics of these are

```
Reg1 ← contents(Reg1) + contents(memory_cell)
Reg1 ← contents(Reg1) + contents(Reg2)
```

The VAX addition instruction for 32-bit integer values is

```
ADDL  operand_1, operand_2
```

whose semantics is

```
operand_2 ← contents(operand_1) + contents(operand_2)
```

Assembly language for IBM mainframe

VAX series minicomputer

Hence VAX language is considered to be more orthogonal.

2. Writability

- *Simplicity & Orthogonality*
- *Abstraction*
- *Expressivity*

3. Reliability

- *Type checking*
- *Exception handling*
- *Aliasing*
- *Readability & Writability*

4. Cost

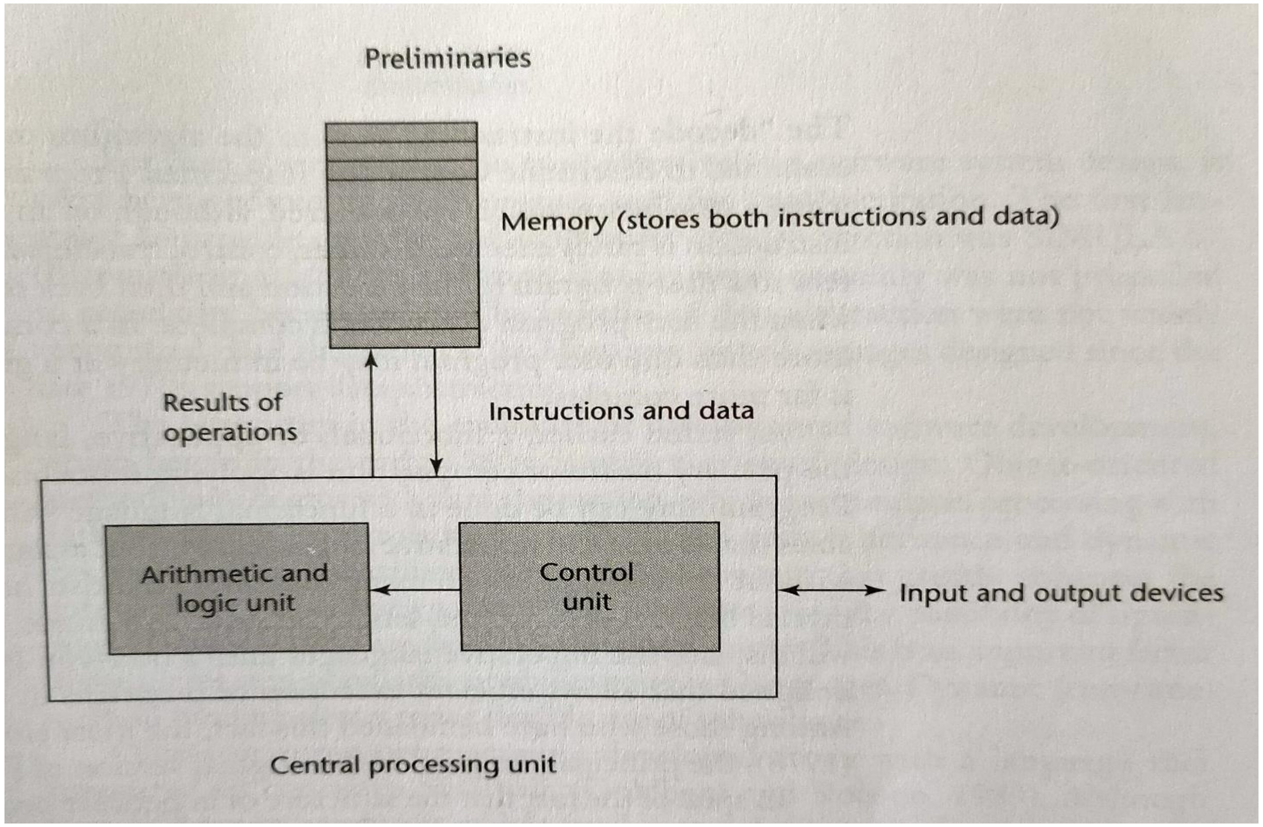
- *Training*
- *Creating SW*
- *Compilation cost*
- *Execution cost*
- *Language implementation system*
- *Poor reliability*
- *Maintenance*

What influence Language Design

1. Computer Architecture
2. Programming Design Methodologies

1. Influence of Computer Architecture on Language design

- Von Neumann Architecture (since 1940)



2. Programming Design Methodologies

- Inadequacies in type checking, use of *goto*.
- 1970 data-oriented or OO paradigm- needs support for important concept called abstraction.
- Need for Concurrent programming

Language categories

1. Imperative Languages (Procedure-oriented)
2. Functional Languages
3. Logic Languages
4. OOP Languages
5. Scripting Languages
6. Mark-up Languages

Language Design trade-offs

1. Writability vs. Reliability (ex. Pointers, large no of operators)

2. Reliabilty vs. cost of execution (ex. Array bound check)

3. Compilation cost vs. execution cost

Language Implementtaion Methods

1. Compilation
2. Interpretation
3. Hybrid approach
4. Pre Processors

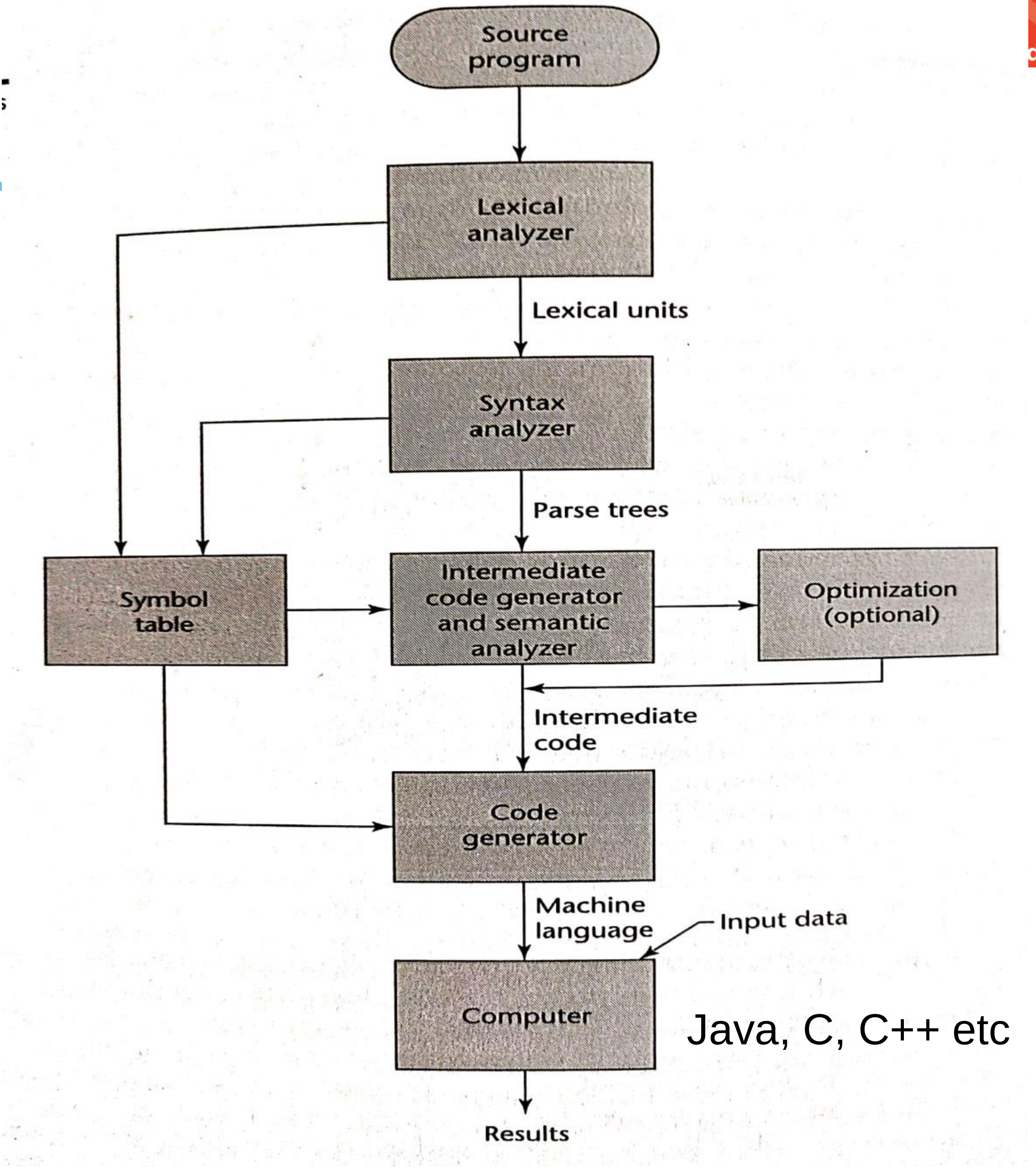
Language Processors

Compiler: It is a program that reads a program written in one language (source language) and translates it into an equivalent program in another language(target language).

It also reports errors in the source program.

It is a Software system for translation.

Compilation



Java, C, C++ etc

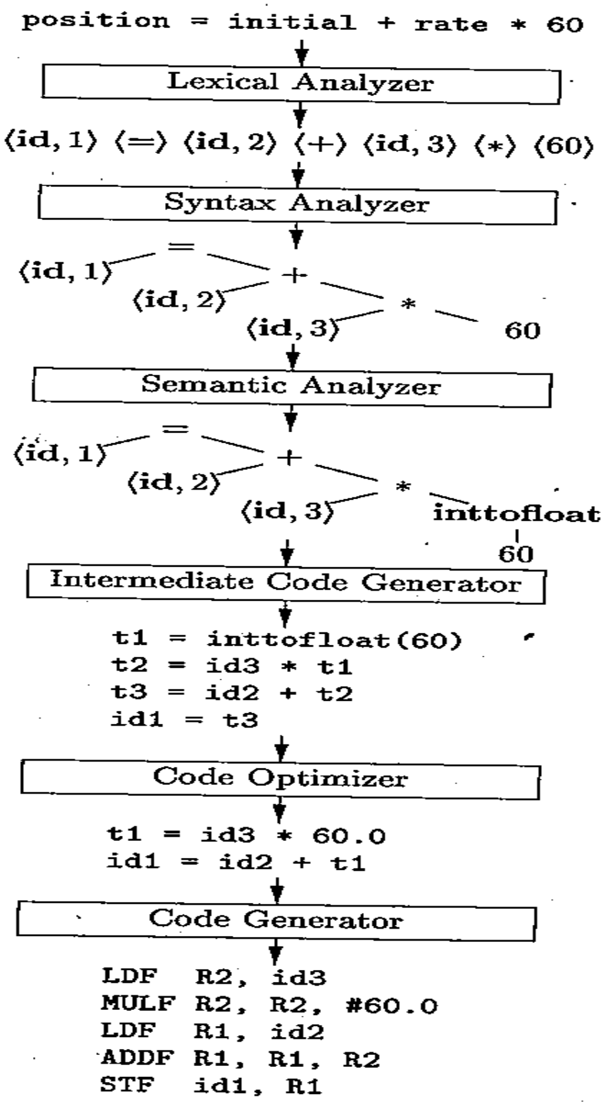
key

<token-name, value>

1	position	...
2	initial	...
3	rate	...

SYMBOL TABLE

<id,1>
↓
symbol table entry



```
t1 = inttofloat(60)
t2 = id3 * t1
t3 = id2 + t2
id1 = t3
```

```
t1 = id3 * 60.0
id1 = id2 + t1
```

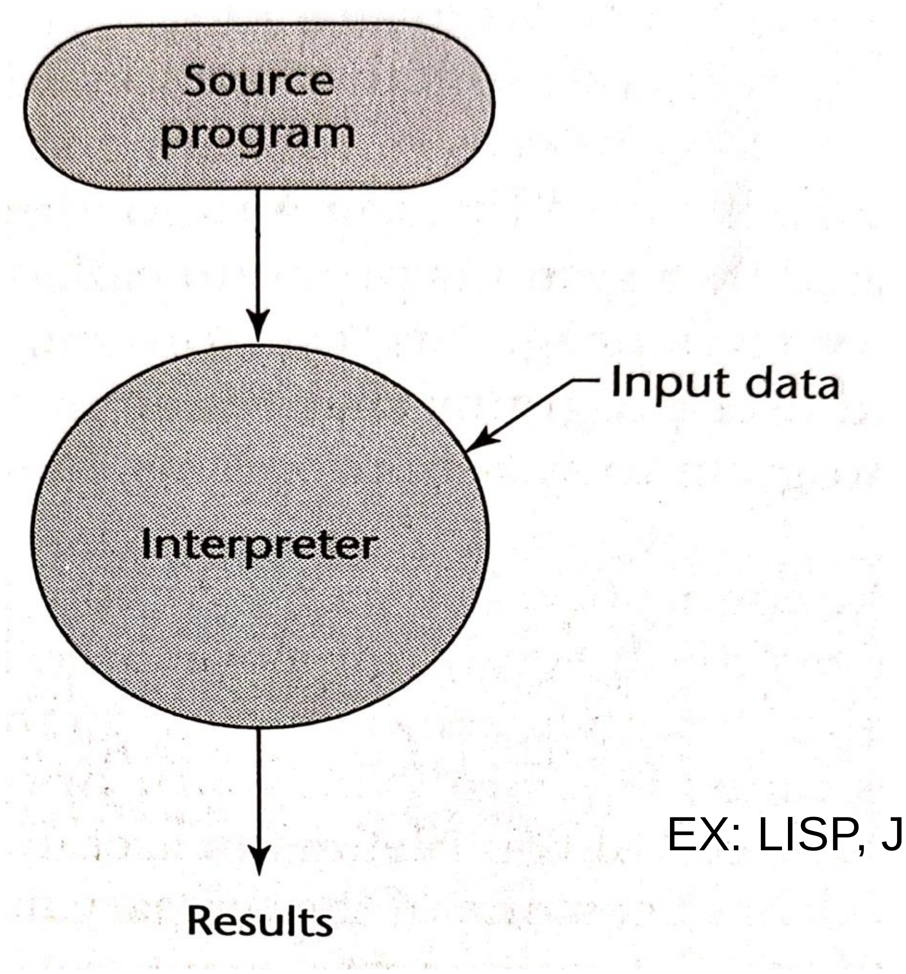
```
LDF  R2, id3
MULF R2, R2, #60.0
LDF  R1, id2
ADDF R1, R1, R2
STF  id1, R1
```

Figure 1.7: Translation of an assignment statement

Language Processors

Interpreter: Instead of producing a target program, it directly executes the operations specified in the source program on inputs supplied by the user and produces output.
With this approach, the programs are interpreted by another program called an interpreter. No translation required.
The interpreter program acts as a SW simulator of machine whose fetch-decode-execute deals with high-level-program statements rather than machine instructions.
This SW simulation provides a VM for the language.
Debugging is easy.
But slow because interpretation must happen every time you execute the program.
A Hybrid approach: In java the byte code is interpreted by JVM.

Interpretation

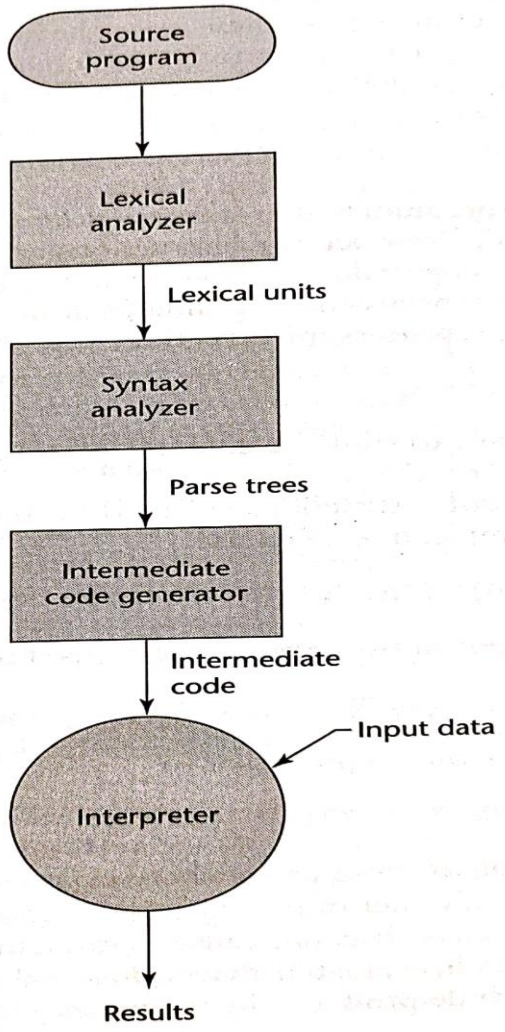


EX: LISP, JavaScript, PHP

Hybrid Approach

Figure 5

Hybrid implementation system



Ex: Java

Programming Environments

A Programming Environment is the collection of tools that help in developing SW. Ex. Editor, Compiler, Linker, Libraries etc.

1. Jbuilder
2. .NET
3. UNIX
4. NetBeans
5. Eclipse

Ch.1 Summary

1. Why study PPL?
2. Programing Domains.
3. Language Evaluation Criteria.
4. Influences on Language Design
5. Language Categories.
6. Design Trade-offs.
7. Implementation Methods.
8. Environments