DESIGN AND IMPLEMENTATION OF PROGRAMMING LANGUAGES

Programming Language

- Machine Language
- Assembly Language
- Define Programming Language?
- Define Abstraction?
- What are the advantages of Abstraction?
- What are the disadvantage of assembly language?

Programming Language

- Fortran
- Algol-60
- State Moore's law?

High-level Language

```
temp   = v[k];
v[k]   = v[k+1];
v[k+1] = temp;
```

```
TEMP   = V(K)
V(K)   = V(K+1)
V(K+1) = TEMP
```

C/Java Compiler

Fortran Compiler

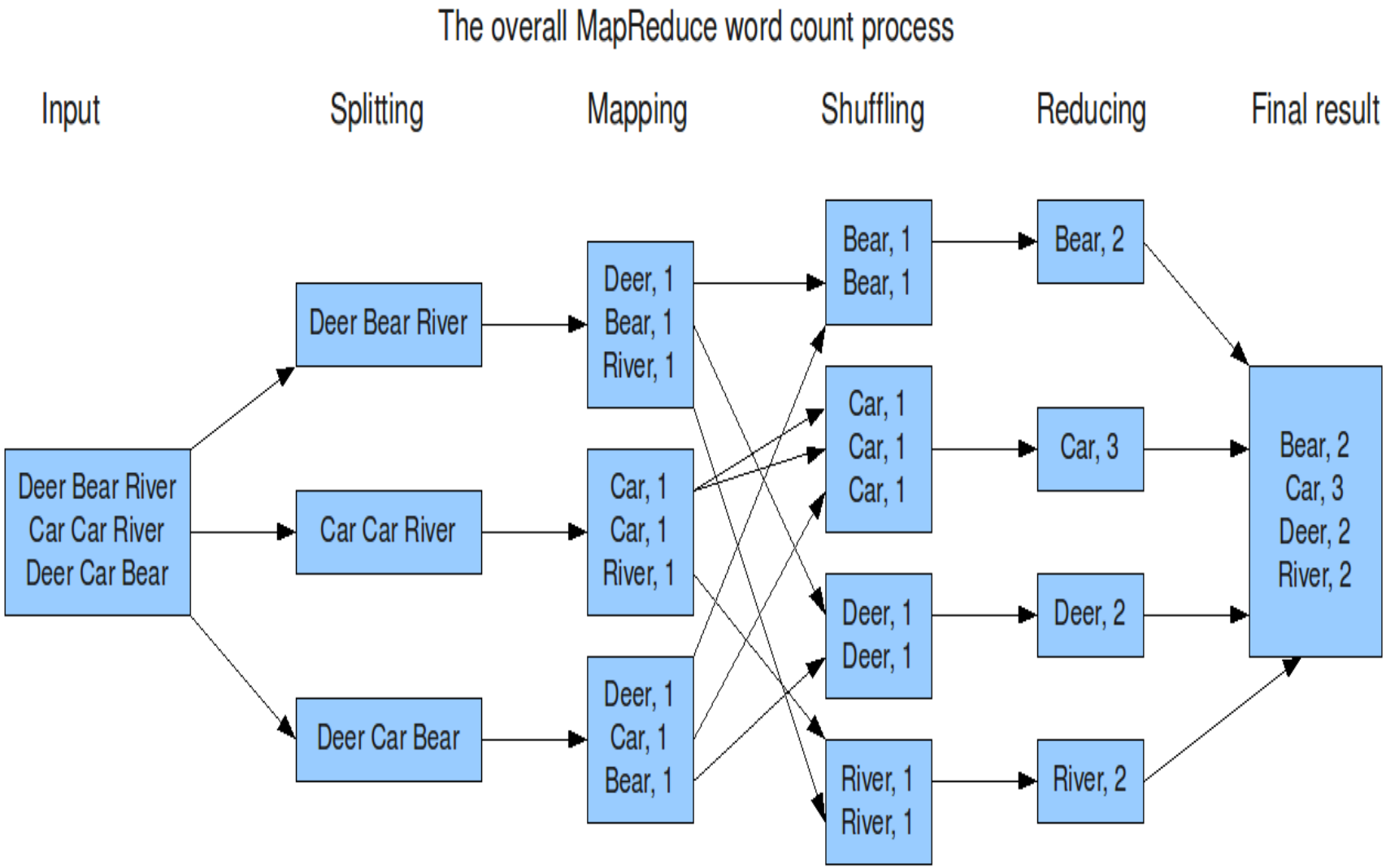
Assembly Language

```
lw  $t0,  0($2)
lw  $t1,  4($2)
sw  $t1,  0($2)
sw  $t0,  4($2)
```

MIPS Assembler

Machine Language

```
0000 1001 1100 0110 1010 1111 0101 1000
1010 1111 0101 1000 0000 1001 1100 0110
1100 0110 1010 1111 0101 1000 0000 1001
0101 1000 0000 1001 1100 0110 1010 1111
```


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

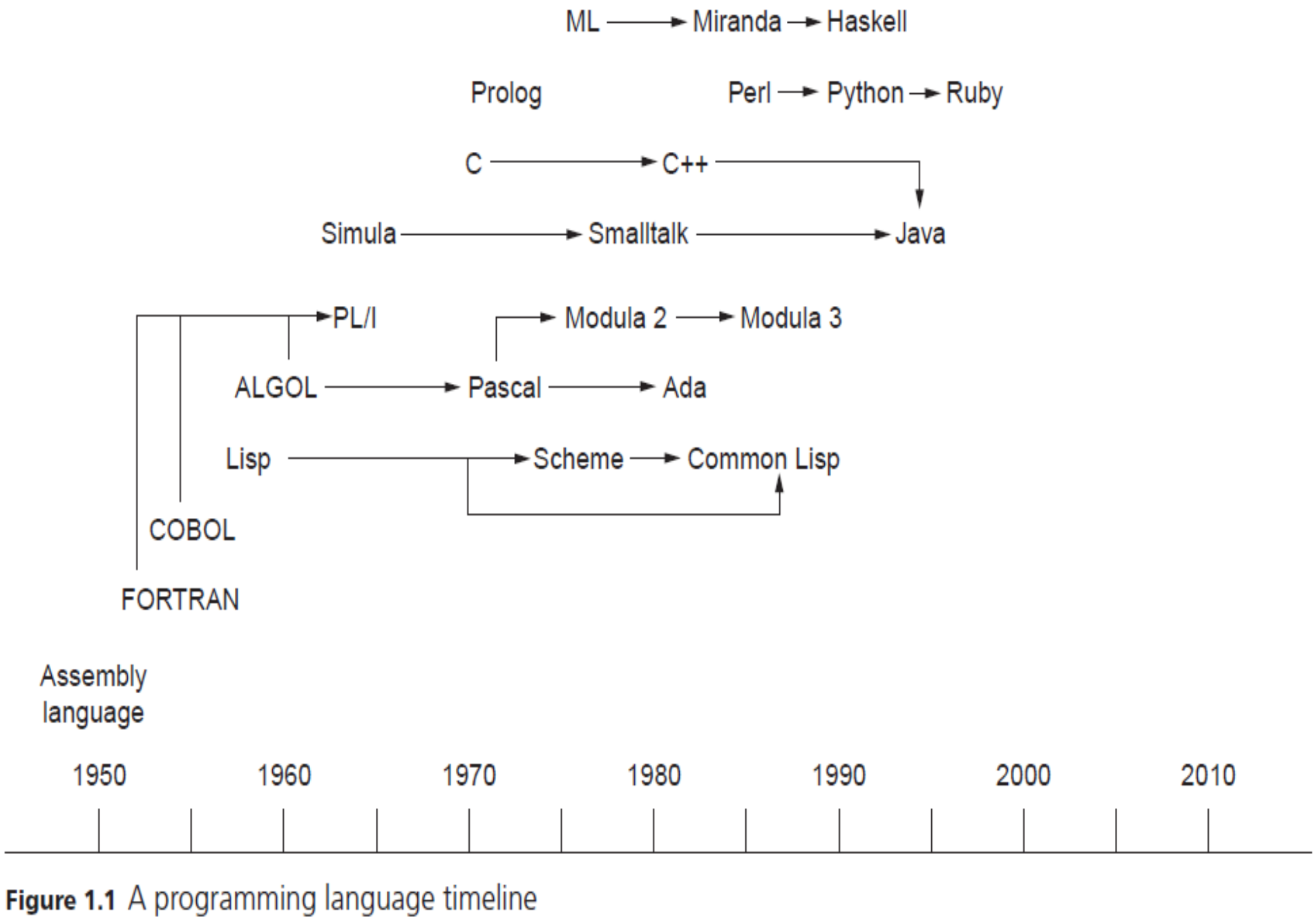


Figure 1.1 A programming language timeline

Introduction

- Programming languages are medium through which we communicate with the computers.
- The languages we discuss in the course are C, C++, Java, Scheme and Prolog

What is a programming language?

- A programming language is a notational system for describing computation in human readable and machine readable form.

What is computation?

- Computation is any process carried out by computer.
- Computation includes all kinds of computer operations including data manipulation, text processing and information storage and retrieval.

What is machine readability?

- There must be some compiler or interpreter which translates the language.
- Compiler/interpreter must not be too complex.
- Machine readability is ensured by restricting the structure of programming language to that of context free languages.

What is Human readability?

- Programming language must be easy to understand.
- Programming languages must be easy to maintain.

Abstraction in programming languages

- Data abstraction
- Control abstraction

Levels of abstraction

- Basic abstraction
- Structured abstraction
- Unit abstraction

Basic abstractions

- Collect together the most localized machine information.

Structured abstraction

- Collect more global information about the structure of the program.

Unit abstractions

- Collect information about entire pieces of program.

Basic data abstraction

- Locations in computer memory that contain data value are abstracted by giving them a name called variables.
- The kind of data value is also given a name and is called a data type.
- Eg:- int x;

Structured data abstraction

- The collection of related data values are abstracted thru data structures.
- Eg:-arrays, structures.
- Many languages allow type declaration
- Eg:-typedef int Intarray[10];
- Such data types are called structured types.

Unit data abstraction

- In a large program it is useful to collect related code into specific locations within a program either as a separate file or as a separate language structure within a file.
- Eg:-package in java, class in C++
- The most important property of unit data abstraction is its reusability.
- Unit abstraction become basis for language library mechanisms

Basic control abstractions

- Statements which combine a few machine instructions into a more understandable abstract statement.
- Eg:-goto

Structured control abstractions

- Structured control abstractions divide a program into groups of instructions that are nested within tests that govern their execution.
- Eg:-if, switch case, while, for, do while, subroutine,(procedure and function)
- Function return value while procedures do not.

Unit control abstraction

- Focus on operation rather than the data.
- The goal is to achieve reusability and library building.

Parallel abstractions

- Allow parallel execution of parts of program.
- Eg:- threads in Java

Turing Completeness

- A language is said to be turing complete when it has…..

  integer variables,

  arithmetic

  and sequentially executes the statements, which includes assignment, selection and loop statements.

Turing completeness

- A programming language is turing complete if it has….

  integer values,

  arithmetic functions on those values, and if it has a mechanism for defining new functions using existing functions, selection, and recursion.

Computational paradigms

- Imperative languages
- Functional languages
- Logical languages
- Object oriented languages
- Parallel

Imperative language

- Imperative language is characterized by following properties.
- The sequential execution of instruction
- The use of variables
- Use of assignment to change the values of variables.
- Eg:-C, Pascal, Ada, Fortran

Imperative language

```
Int fact(int n)
{
        int sofar=1;
        while(n>0)
        sofar *=n--;
        return sofar;
}
```

Hall marks of Imperative languages

- Assignment
- Iteration
- Order of execution is critical

Functional languages

```
(define fact(x) (if (<= x 0) 1(* x(fact(- x 1)))))
```

Hall mark of functional languages

- Single valued variables : no assignment
- Heavy use of recursion : no iteration

Logical languages

```
fact(x,1):-x:=1
fact(x,Fact):-x>1, newx is x-1,
                   fact(newx,nf),
                   Fact is x*nf
```

Hallmark of logic languages

- Program expressed as rules in formal logic.

Object Oriented Languages

```
class Factorial
{
        int facto(int n)
        {
                int fac=1;
                for(int i=n;i>0;i--)
                {
                        fac*=i;
                }
                return(fac);
        }
        public static void main(String args[])
        {
                Factorial f=new Factorial();
                int fact=f.facto(5);
                System.out.println("Factorial of 5="+fact);
        }
}
```

Hallmarks of object oriented languages

- All properties of imperative plus …
- objects

Pure languages

- Imperative:- (old)fortran
- Functional:- haskell
- Object oriented:- small talk

Question

- Can you execute statement within an if statement without checking its condition in C/C++ and in JAVA why or why not

Permissive Languages

```
Goto my10;
i=5;
If(i==12)
{
  my10:cout<<“I can reach here in C++”;
}
```

Language definition

- Rules that govern the language
- American National Standard Institute (ANSI)
- International Organization for Standardization (ISO)

Language definition

- Language definition can be divided into two parts
- Syntax (structure)
- Semantics (meaning).

Language Syntax

- The syntax of the programming language is like grammar of a natural language.
- Syntax of all programming languages is given using context-free grammars.
- if-statement→if(expression) statement [else statement]

Lexical structure

- The structure of words or tokens.
- Eg:-if, else, identifiers, +, ; etc

Language Semantics

- The actual result of execution.
- Usually described in English but can be done mathematically.
- Semantics can have static components such as type checking, definition checking other consistency checks prior to execution.

Semantic of If statement

- An if statement is executed by first evaluating its expression, which must have arithmetic or pointer type, including all side effects, and if it compares unequal to zero, the statement following the expression is executed. If there is an else part, and the expression is zero, the statement following the else is executed.

Language Translation

- Compiler:- Two step process that translate source code into target code then user executes the target code.
- Interpreter:-one step process where the source code is executed directly.
- Java code is compiled to a machine independent set of instructions called byte code. Byte code is then interpreted by the Java Virtual Machine(JVM).

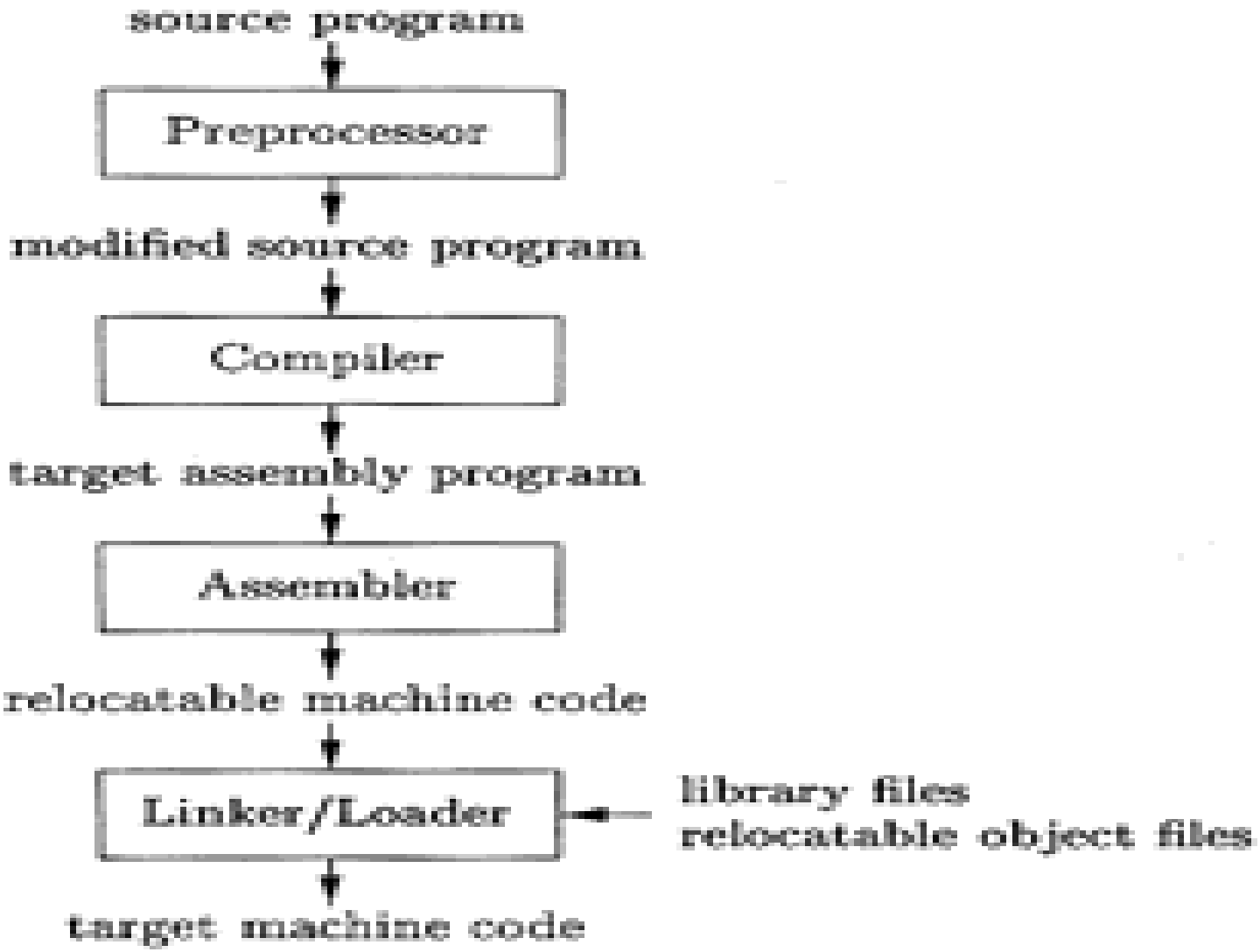


Figure 1.5: A language-processing system

Phases of translation

- Source program is converted into tokens by lexical analyzer.
- Tokens are converted into structure of the program by syntax analyzer.
- Structure of the program is converted into proper meaningful program by semantic analyzer.

Error classification

- Lexical
- Syntax
- Static semantic
- Dynamic semantic
- logical

Lexical error

- Character level error such as illegal character.
- Eg:- abc#, ab$

Syntax error

- Error in structure
- Eg:- missing semicolon or keyword

Static semantic

- Non syntax error prior to execution
- Eg:-Undefined variables, incompatible type.

Dynamic semantic

- Non syntax error during execution.
- Eg:- division by zero.

  array index out of range.

Logical errors

- Programmer errors.
- Logic errors are not errors at all from the point of view of language translation.

Error reporting

- A compiler will report lexical, syntax and static semantic errors. It cannot report dynamic sematic errors.
- An interpreter will only report lexical and syntax errors when loading the program. Static semantic errors may not be reported until just prior to execution.
- No translator can report a logic error.

Run time environment

- During execution language translators must maintain run time environment and allocation of memory to program and data.

Run time environment

- An interpreter maintains Run time environment internally as a part of its management of the execution of the program.

Run time environment

- Compiler adds suitable instructions to the target code to maintain run time information.

Birth of languages

- Unix→C
- Internet→java
- OOApproaches→C++.

Imperative style

```
int numdigits(int x)
{
        int temp=x; n=1;
        while(temp>=10)
        {
                n++;
                temp=temp/10;
        }
        return n;
}
```

Functional style

```
int numdigits(int x)
{
        if(x<10) return 1;
        else
        return 1+numdigits(x/10);
}
```

Question

- Write down the while statement equivalent of following statement

  if(e) s1 else s2

Answer

```
X=e;
Y=1-X;
while(X)
{
        s1;
        X=0;
}
while(Y)
{
        s2;
        Y=0;
}
```

Question

- Write a program in C/C++ to reverse a string using functional paradigm(i.e. using recursion).

```
# include <stdio.h>
void reverse(char *str)
{
   if(*str)
   {
      reverse(str+1);
      printf("%c", *str);
   }
}

int main()
{
   char a[] = “MIT";
   reverse(a);
   getchar();
   return 0;
}
```

Fibonacci Iterative

```
static int FibonacciIterative(int n)
{
    if (n == 0) return 0;
    if (n == 1) return 1;

    int prevPrev = 0;
    int prev = 1;
    int result = 0;

    for (int i = 2; i <= n; i++)
    {
        result = prev + prevPrev;
        prevPrev = prev;
        prev = result;
    }
    return result;
}
```

Scheme Program

```
(define (fib n)
 (cond((= n 0) 0)
        ((<= n 2) 1)
 (else (+ (fib (- n 1)) (fib (- n 2))))
 )
 )
```

Fibonacci in Prolog

```
fibonacci(0,0).
fibonacci(1,1).
fibonacci(2,1).
fibonacci(N,F) :- if (N>=3)
        fibonacci(N-1,F1), fibonacci(N-2,F2), F
= F1 + F2.
```

References

Text book

- Kenneth C. Louden and Kenneth Lambert “Programming Languages Principles and Practice” Third edition Cengage Learning Publication.

Reference Books:

- Terrence W. Pratt, Masvin V. Zelkowitz “Programming Languages design and Implementation” Fourth Edition Pearson Education.
- Allen Tucker, Robert Noonan “Programming Languages Principles and Paradigms second edition Tata MC Graw –Hill Publication.