GLOBAL EDGE
Intelligence Of Things

COMPILATION PROCESS

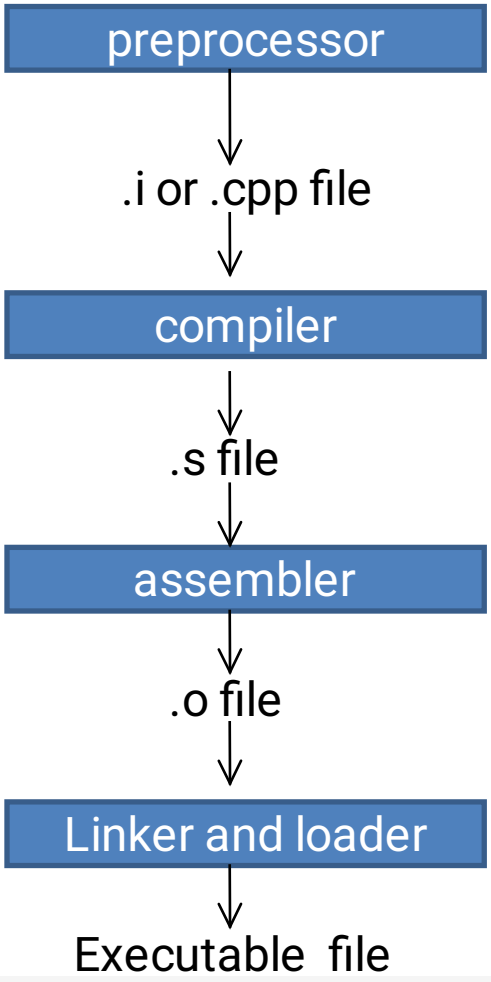
MUTHU GOMATHY V
KOMMAGONI ESWAR BABU

GLOBAL EDGE

CONTENTS

- PREPROCESSOR
- COMPILER
- ASSEMBLER
- LINKER
- LOADER

Compilation process



GLOBAL EDGE
Intelligence Of Things

GLOBAL EDGE

C preprocessor

GLOBAL EDGE
Intelligence Of Things

- What is C preprocessor?
- Preprocessing directives.
- Header file inclusion.
- Macros.
- Comment removal.
- Conditional compilation.

What is c preprocessor?

GLOBAL EDGE
Intelligence Of Things

The C preprocessor is a macro proprocessor that is used automatically by C compiler to transform your program before actual compilation.

The C preprocessor provides four separate facilities.

- Header file inclusion
- Macro expansion
- Comment removal
- Conditional compilation

Preprocessing Directives

GLOBAL EDGE
Intelligence Of Things

- All Preprocessor directives start with '**#**'.
- The '**#**' is followed by an identifier that is the directive name.
- The set of valid directive names are fixed.
- Programs cannot define new preprocessing directives.
- Some directive names require arguments; these make up the rest of the directive line and must be separared from the directive name by whitespace.
- A preprocessing directive cannot be more than

Common preprocessor directives

GLOBAL EDGE
Intelligence Of Things

#define - create a macro

#undef - remove a macro

#include - replace the current line with the contents of a file

#ifdef - compile code if a macro has been defined with a previous **#define**

#if - compile code if an expression is true

#else - compile code if a previous **#if** or **#ifdef** failed

#endif - terminate a previous **#if**,**#ifdef** or **#else**

Cont..

GLOBAL EDGE
Intelligence Of Things

- C comments and predefined macro names are not recognized inside a '**#include**' directive in which the file name is delimited with '<' and '>'.

Example: **#include</**/stdio.h>**

- C comments and predefined macro names are never recognized within a character or string constant.

Example 1: **'#include<stdio.h>'**

Example 2: **"#include<stdio.h>"**

Header File inclusion

GLOBAL EDGE
Intelligence Of Things

- A header file is containing C declarations and macro definitions to be shared source files.

- Including a header file produces the same results in compilation as copying the header file into each source file that needs it.

GLOBAL EDGE

- Header files serve two kinds of purposes.

  System header files.

Cont..

GLOBAL EDGE
Intelligence Of Things

- Both user and system header files are included using the preprocessing directive **'#include'.** It has two variants.

1) #include <file>:

- This variant is used for system header files. It searches for a file named "file" in a list of standard system directories.

2) #include "file":

GLOBAL EDGE

- This variant is used for header files of your own program.

Macros

GLOBAL EDGE
Intelligence Of Things

- A macro is a sort of abbreviation which you can define once and then use later.
- There are many complicated features associated with macros in the C preprocessor.

1)Simple Macros

Example: **#define** max 1

GLOBAL EDGE

2)Macros with Arguments

Example: **#define** max(x,y) x > y ? x : y

3)Undefining macros

Cont..

GLOBAL EDGE
Intelligence Of Things

4) Redefining macros

Redefining a macro means defining a name that is already define as a macro.

Some of the Predefined macros

__FILE__

__DATE__

__TIME__

__LINE__

Cont..

GLOBAL EDGE
Intelligence Of Things

- We can concatinate two strings using '**##**'

Example: **#define** cat(a,b) a##b

```
int main()
{
        printf("%d",cat(12,34);
        return 0;
}
```


GLOBAL EDGE

Cont..

GLOBAL EDGE
Intelligence Of Things

- The C preprocessor doesn't understand the syntax or precedence.
- Several bugs can appear because of this lack.

Example :

```
#define square(i) i * i

Square(2+3)
```


GLOBAL EDGE

Here the macro will be expand as 2+3*2+3

so be careful when you use macros

Comment removal

GLOBAL EDGE
Intelligence Of Things

- All C comments are replaced with single spaces.
- Backslash-Newline sequences are deleted,no matter where.
- The two transformations are done before nearly all other parsing and before preprocessing directives are recognized.

Conditional compilation

GLOBAL EDGE
Intelligence Of Things

- A conditional is a directive that allows a part of the program to be ignored during compilation, on some conditions.

Why conditionals are used?

- A program may need to use different code depending on the machine or operating system it is to run on. In some cases the code for one operating system may be erroneous on another operating system.
- This case conditionals are used.
- Most simple programs that are intended to run on only one machine will not need to use preprocessing conditionals.

Cont..

GLOBAL EDGE
Intelligence Of Things

- Conditionals directives are '**#if**','**#ifdef**','**#ifndef**','**#else**','**#elif**','**#endif**'

'**#if**' directive in its simplest form consists of

#**if** expression

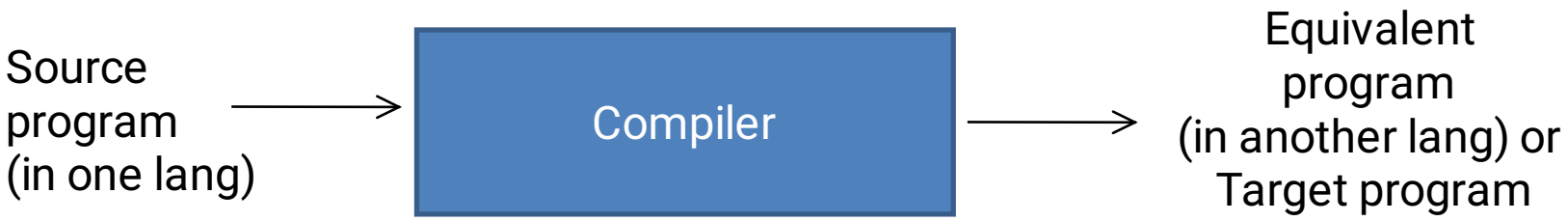
    controlled text

#**endif**

you can put anything at all after the '**#endif**' and it will be ignored by the preprocessor.

GLOBAL EDGE
Intelligence Of Things

WHAT IS A COMPILER ?



GLOBAL EDGE
Intelligence Of Things

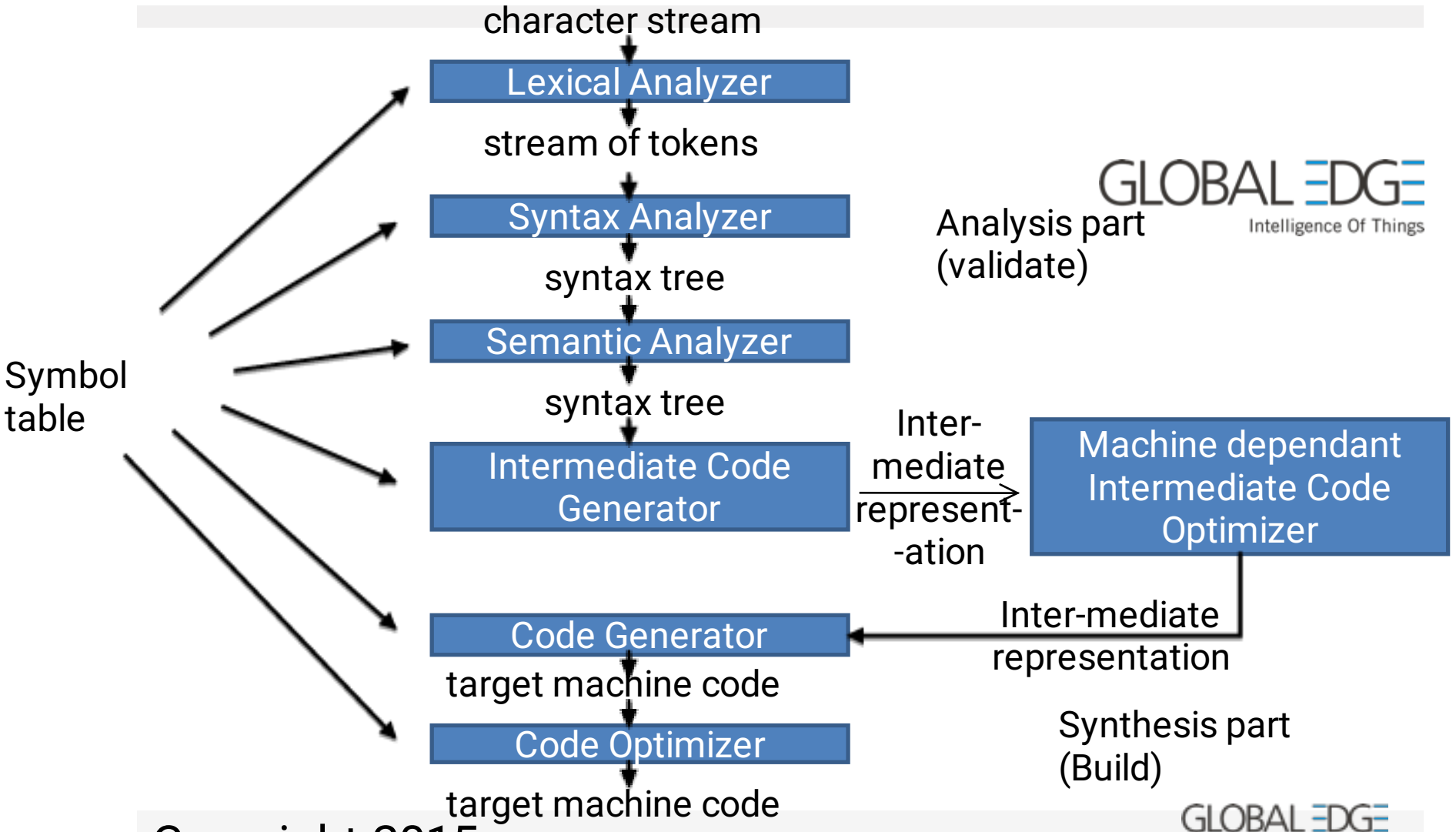
Why Compiler ?

GLOBAL EDGE

ROLE OF A COMPILER :

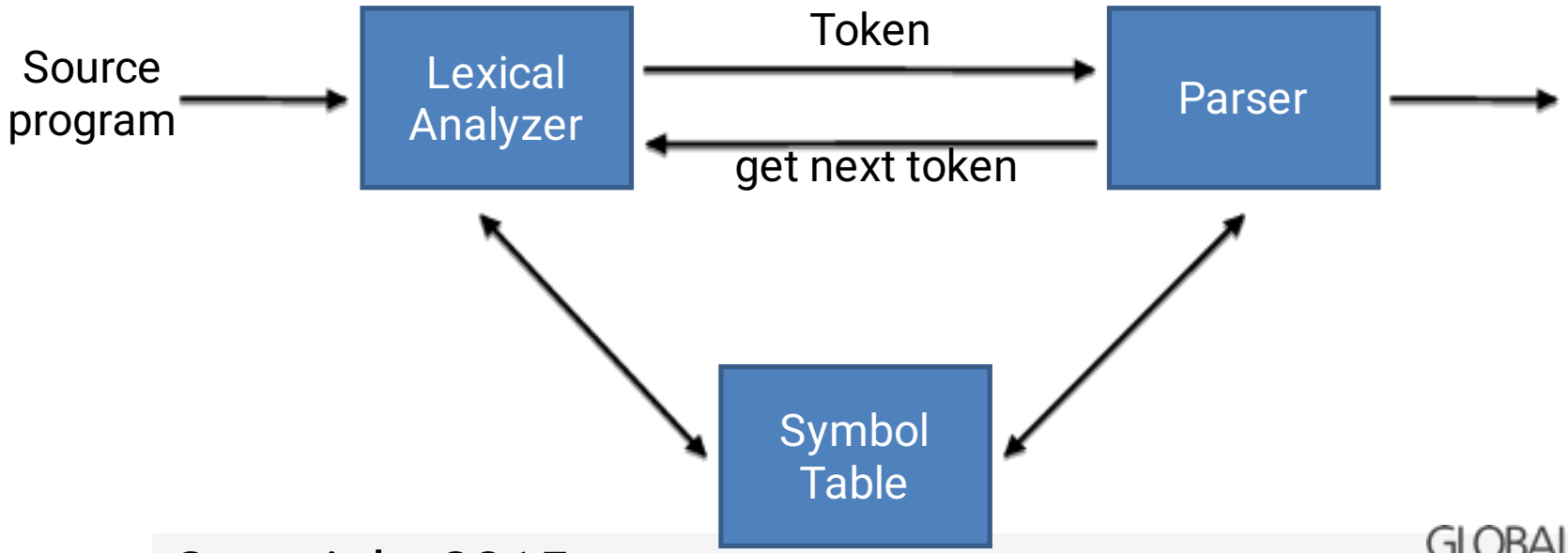
GLOBAL EDGE
Intelligence Of Things

- Report any errors in the source program
- Optimizing the program
- Generating the intermediate code which can be used to generate a different machine codes for different machines



Lexical analyzer

- First phase of a compiler
- Reads the stream of characters making up the source program and groups the characters into meaningful sequences called *lexemes.*



3 terminologies

Token : a set of input string which are related through a similar pattern

Lexeme - the input string that represents a valid token is a lexeme

Pattern – the rule that says that it is avalid token is a pattern.

GLOBAL EDGE
Intelligence Of Things

Example :
int abcd
Token 1: Identifier , keyword
Lexeme : int , abcd
Pattern : Rule that says that it is an keyword

Lexical Grammar : for identifier
Letter : _[a-z][A-Z]
Digit : [0-9]
Regular Expression for pattern:
id -> letter (letter | digit)*

Types of tokens

Identifiers	id, val,val123
Keywords	int, return
Constants	123, 03, 0x123, 'a'
String literals	”hello”
Operators	+, -, *, \, & ,| ,!=,
Separators	, ( ) [ ] ;

GLOBAL EDGE
Intelligence Of Things

- The process of forming tokens from an input stream of characters is called **tokenization.**
- Also called as a **lexer**, **tokenizer**, or **scanner** .

GLOBAL EDGE

GLOBAL EDGE
Intelligence Of Things

/ , *, = , +, -
/ * , == , != , >= , ->
identifiers
Keywords
constants

GLOBAL EDGE
Intelligence Of Things

How tokenization is done ?

“If the input stream has been parsed into tokens up to a given character, the next token is taken to include the longest string of characters which could possibly constitute’ a token.”

GLOBAL EDGE

GLOBAL EDGE
Intelligence Of Things

How recognition of a token is done ?

GLOBAL EDGE

For each lexeme, the lexical analyzer produces a output a *token of the form*

GLOBAL EDGE
Intelligence Of Things

(token-name, attribute-value)

abstract symbol used during syntax analysis

points to an entry in the symbol table for this token

Example :

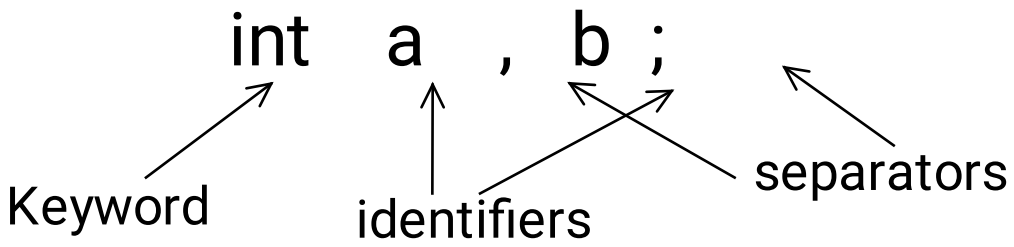
sum = val_1 + val_2 * 20

Sum : (id,1)
= : (=)
Val_1 : (id,2)
+ : (+)
Val_2 : (id,3)
* : (*)
20 : (20)

(i d , 1) (=) (id, 2) (+) (id, 3) (*) (60)

Valid

GLOBAL EDGE
Intelligence Of Things



Invalid

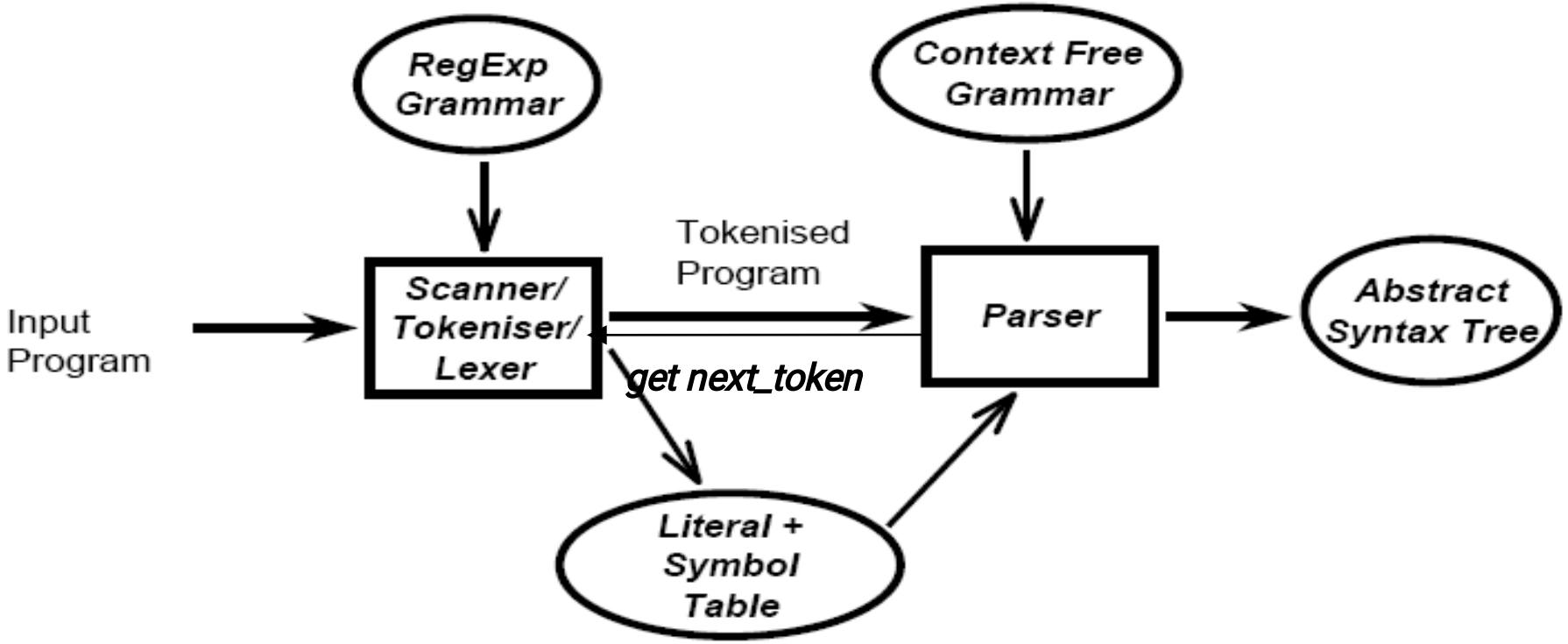
Characters not valid in C :

@ (at the rate of)
` (apostate key)

Eg: `h` instead of ‘h’

```
int 2ab; //invalid suffix ab on an integer constant
int a=2ab; //invalid suffix ab on an integer constant
int  a=0xg;  //invalid sufix to a hexadecimal integer
int b=09; //invalid use of octal integer
```

Syntax analysis



Role of a parser

- Performs context-free syntax analysis
- Guides context-sensitive analysis
- Constructs an intermediate representation
- Produces meaningful error messages
- Attempts error correction

GLOBAL EDGE
Intelligence Of Things

4 components of CFG :

A set of ,

1. Terminals.

2. Nonterminals

3. Productions,

4. Start symbol.

context-free grammars

GLOBAL EDGE
Intelligence Of Things

if (expression) statement else statement

stmt —> if (expr) stmt else stmt

This arrow may be read as "**can have the form**."
Such a rule is called a production.

keyword **if** and the **parentheses** are called *terminals.*
Variables like ***expr*** and ***stmt*** represent sequences of terminals and are called *nonterminals.*

GLOBAL EDGE
Intelligence Of Things

Language :

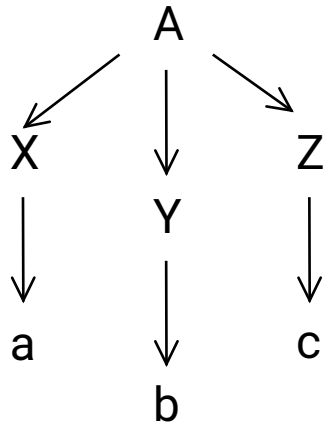
- A grammar derives strings by beginning with the start symbol and repeatedly replacing a nonterminal by the body of a production for that nonterminal.
- The terminal strings that can be derived from the start symbol form the language defined by the grammar.

Example : {pos=cnt+1; , if(i>1) i=i; else i=0; }

Parse tree

A -> XYZ
X->a
Y->b
Z->c

String : abc



Properties:
1. The root is labeled by the start symbol.
2. Each leaf is labeled by a terminal or by e.
3. Each interior node is labeled by a nonterminal. 4. If *A is the nonterminal labeling some interior node and X1, X2, • • • , Xn are* the labels of the children of that node from left to right, then there must be a production *A —> X1X2 • • • Xn. Here, X\,X2,.. • ,Xn each stand* for a symbol that is either a terminal or a nonterminal. As a special case, if *A ->• e is a production, then a node labeled A may have a single child* labeled e.

Constructing a parse tree the string :

3+4*5

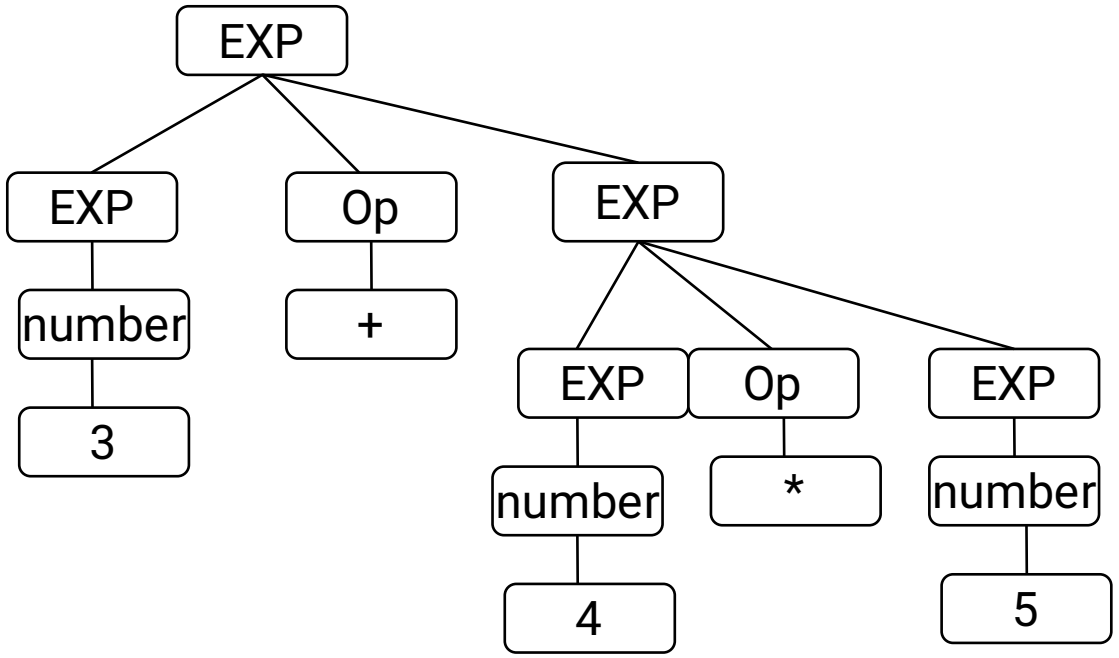
The grammar will be :

EXP -> EXP + EXP | EXP – EXP | EXP * EXP

EXP -> number

number -> [0-9]

GLOBAL EDGE
Intelligence Of Things



GLOBAL EDGE
Intelligence Of Things

What is Parsing ?

GLOBAL EDGE

GLOBAL EDGE
Intelligence Of Things

The process of finding a parse tree for a given string of terminals is called parsing that string.

GLOBAL EDGE

GLOBAL EDGE
Intelligence Of Things

Why Parsing ?

GLOBAL EDGE
Intelligence Of Things

Parsing is the problem of taking a string of terminals and figuring out how to derive it from the start symbol of the grammar, and if it cannot be derived from the start symbol of the grammar, then reporting syntax errors within the string. Parsing is one of the most fundamental problems in all of compiling;

GLOBAL EDGE

Methods of parsing

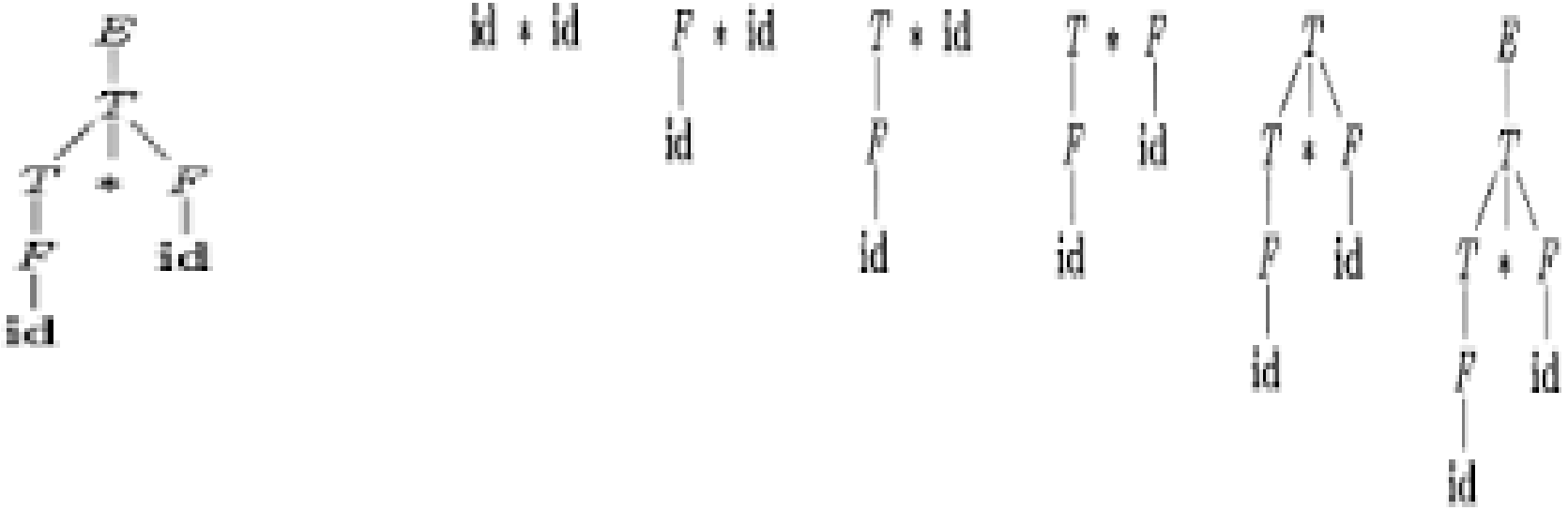
Top-down parser

Starts - at the root and
Proceeds - towards the leaves,

Bottom-up parser

GLOBAL EDGE
Intelligence Of Things

Starts - at the leaves and
Proceeds- towards the root



GLOBAL EDGE

Syntax tree

represents the hierarchical syntactic structure of the source program.

GLOBAL EDGE
Intelligence Of Things

E1 op E2

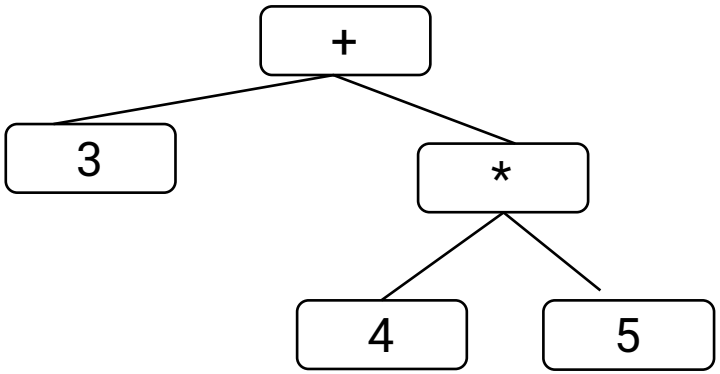
op
/ \
E1 *E2*

- *each* interior node -> is an operator
- the children of the node -> the operands of the operator.

GLOBAL EDGE

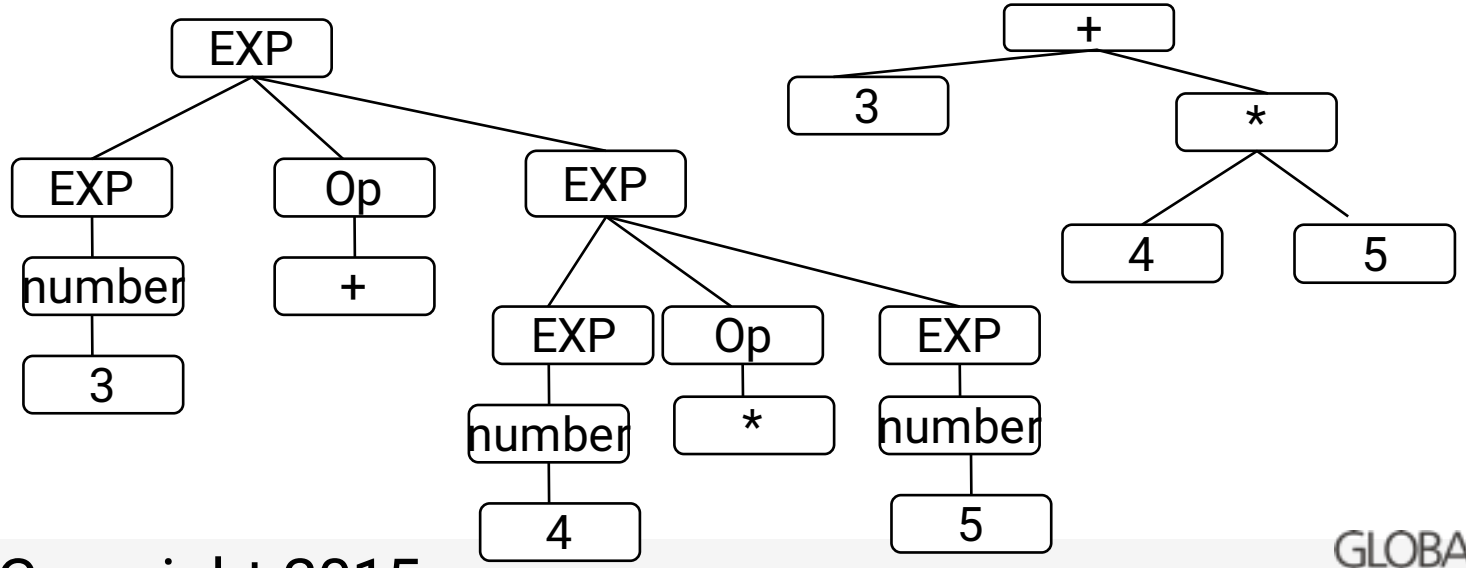
Example for syntax tree

3+4*5



Difference between the parse tree and the syntax tree

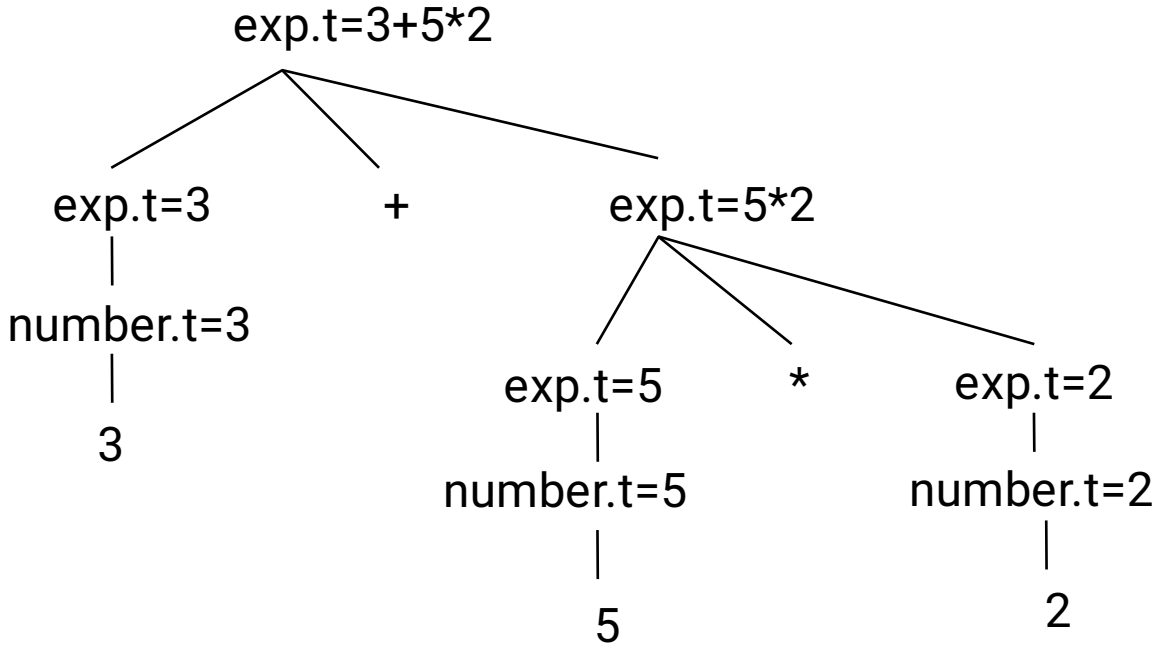
	syntax trees	Parse tree
Interior nodes	: represent programming constructs	represent nonterminals.
Leaf node	: operators	terminal symbols
Helpers	: required	not required



Annotated parse tree

EXP -> EXP + EXP | EXP – EXP | EXP * EXP
EXP -> number
number -> [0-9]

GLOBAL EDGE
Intelligence Of Things



Symbol table

```
int a=10;  float b=20.23;
int main() {
int c=30;
int d=40;
}
```

GLOBAL EDGE
Intelligence Of Things

	Variable name	Class	Type	value
1:				
2:	a	fixed	int	10
3:	b	fixed	float	20.23
4:	c	auto	int	30
	d	auto	int	40

GLOBAL EDGE

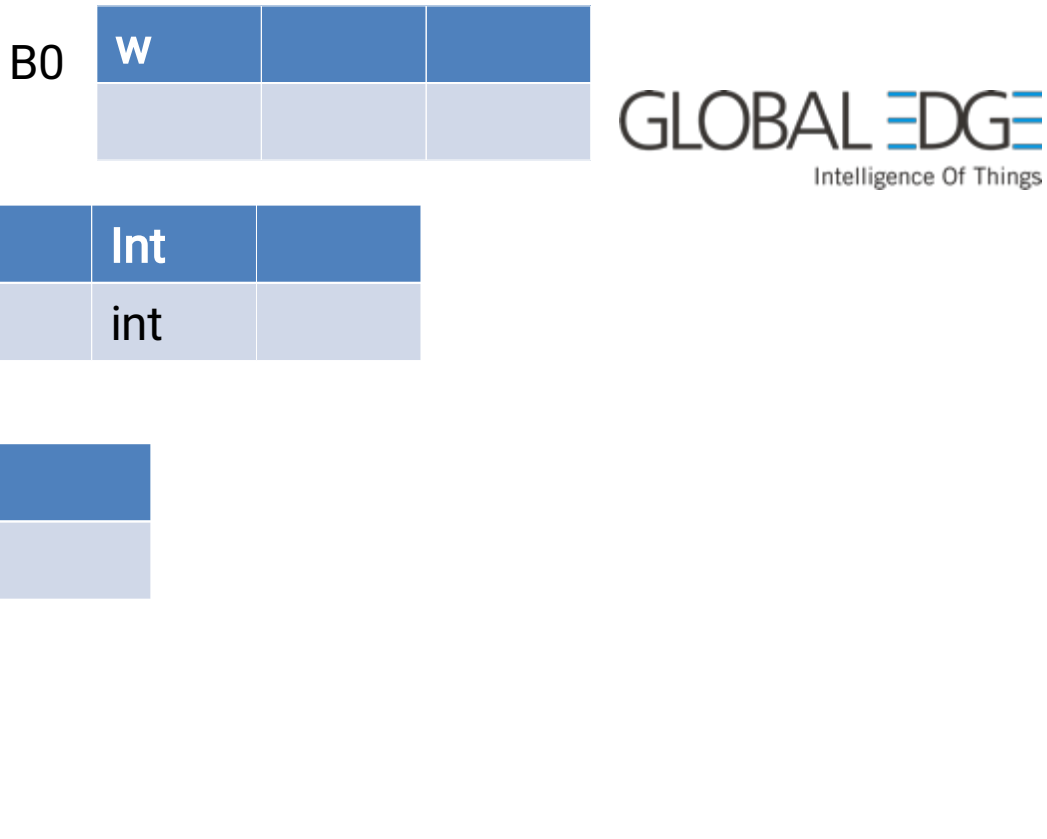
GLOBAL EDGE
Intelligence Of Things

Who Creates Symbol-Table Entries?

GLOBAL EDGE

Optimizing symbol table

```
Int a;
Int b;
Int main()
{
 int c, d;
  {
    int e, f;
   }
 }
```



GLOBAL EDGE
Intelligence Of Things

GLOBAL EDGE

Syntax errors

int ab% //syntax

ab%db //syntax

int ab&ch; //syntax
int a b; //syntax

Ab+2
Int a=a; //globally

A=a+b //missing semicolon
Missing braces // control reaches end of non void function

GLOBAL EDGE
Intelligence Of Things

GLOBAL EDGE

Semantic analysis

GLOBAL EDGE
Intelligence Of Things

- An important part of semantic analysis is type checking, where the compiler checks that each operator has matching operands.

GLOBAL EDGE
Intelligence Of Things

Static semantics checks :

- Declarations of variables and constants before use
- Calling functions that exist (predefined in a library or defined by the user)
- Passing parameters properly
- Type checking.
- Annotates the syntax tree with type information

Static Checking

GLOBAL EDGE
Intelligence Of Things

Syntactic Checking.
type Checking.

GLOBAL EDGE

coercions

A= 2.1* 30;

Semantic ananlysis imposes type checking

A= 2.1 * inttofloat(30);

A=2.1 * 30.0

L-values and R-values

GLOBAL EDGE
Intelligence Of Things

Lvalue		Rvalue
i	=	**5;**
i	=	i + **1;**

GLOBAL EDGE

Semantic errors

GLOBAL EDGE
Intelligence Of Things

5=a;
int 5;

Int a=a; //globally

Int b=0;
Static int a=b;

Intermediate Code Generation

GLOBAL EDGE
Intelligence Of Things

- In the process of translating a source program into target code, a compiler may construct one or more intermediate representations,which can have a variety of forms.
- Synatx trees are a form of intermediate represenrtation, they are commonly used during syntax and semantic analysis.

Simple Example: a = b + c

GLOBAL EDGE

=

/ \

Cont..

GLOBAL EDGE
Intelligence Of Things

Three Address code

In a three address code there is at most one operator at the right side of an instruction.

Forms of Three Address Instructions

→ x = y op z

→ x = op y

→ x = y

Cont...

GLOBAL EDGE
Intelligence Of Things

→ Example for Three Address Code

a = b + c * d;

t1 = c(id3) * d(id4);

t2 = b(id2) + t1;

a(id1) = t2;

→ How to present these instructions in a data structure?

→ Quadruples

→ Triples

Cont..

GLOBAL EDGE
Intelligence Of Things

→ **Quadraples**: it has fourfields:op,operand1,operand2,result

Note: for unary operators there is no second argument

Example: x = (a + b) * -c /d

	operator	operand1	operand2	result
1)	+	a	b	t1
2)	-	c		t2
3)	*	t1	t1	t3
4)	/	t3	d	t4
5)	=	t4		x

GLOBAL EDGE

Cont..

GLOBAL EDGE
Intelligence Of Things

→ **Triples** : only three fields,no result field.results referred to by its position.

	Operator	operand1	operand2
(1)	+	a	b
(2)	-	c	
(3)	*	(1)	(2)
(4)	/	(3)	d
(5)	=	x	(4)

Code optimization

GLOBAL EDGE
Intelligence Of Things

-> levels of optimizations

1) o0 - Default optimizer

2) o1 - It reduces the code size and increases the execution speed.

3) o2 - It increases the execution time and performance.

4) o3 - it is also increases the execution time.

5) os - it reduces the size.

Cont..

GLOBAL EDGE
Intelligence Of Things

→ Common subexpression elimination.

Instructions that compute a value that has already been computed.

→ Dead code elimination.

Instructions that compute a value never used.

→ Loop optimizations

→ invariant code

→ induction analysis

Code Generation

GLOBAL EDGE
Intelligence Of Things

→ Basic blocks comprise of a sequence of three-address instructions. Code generator takes these sequences of instructions as input.

→ The code generator should take the following things into consideration to generate the code:

- Target language
- IR Type
- Selection of instruction
- Register allocation
- Ordering of instructions

Cont..

GLOBAL EDGE
Intelligence Of Things

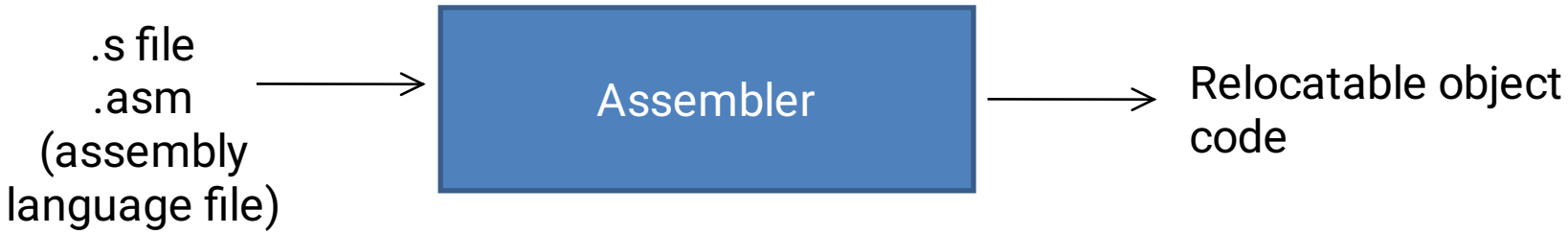
→ Code generator uses **getReg** function to determine the status of available registers and location of name values.

→ It allocate registers and it is translates into assembly language.

→ The assembly lanuage is easily understand by user, because it is same as english.

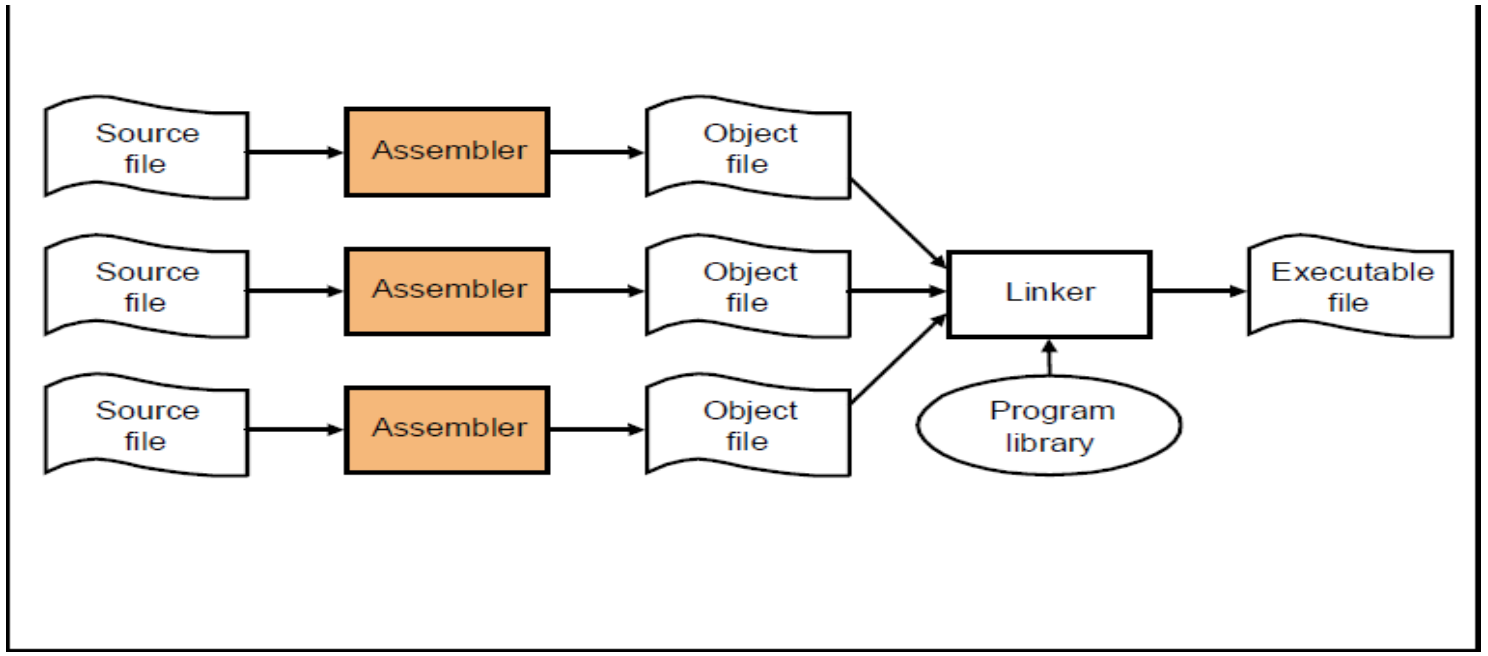
Assember

GLOBAL EDGE
Intelligence Of Things



Linker

GLOBAL EDGE
Intelligence Of Things



Three tasks:-

- Searches the program to find library routines used by program, e.g. printf(), sqrt(),strcat() and various other.
- Determines the memory locations that code from each

module will occupy and relocates its instructions by

adjusting absolute references.

Relocation, which modifies the object program so that it can be loaded at an address different from the location originally specified.

- It combines two or more separate object programs and supplies the information needed to allow references between them .

Linking Concepts

Computer programs typically comprise several parts or modules; all these parts/modules need not be contained within a single **object file**, and in such case refer to each other by means of **symbols**. Typically, an object file can contain three kinds of symbols:`

- ✓ ***Externally defined symbols(***undefined symbols), which calls the other modules where these symbols are defined, also called as **external reference.**
- ✓ ***Local symbols***, used internally within the object file to facilitate relocation.

LINKER ERROR

- undefined reference to a function
- undefined reference to a variable

GLOBAL EDGE
Intelligence Of Things

GLOBAL EDGE

Loader

It is a system program that brings an executable file residing on disk into memory and starts it running.

GLOBAL EDGE
Intelligence Of Things

Steps:
– Read executable file's header to determine the size of text and data segments.

– Create a new address space for the program.

– Copies instructions and data into address space.

– Copies arguments passed to the program on the stack.

– Initializes the machine registers including the stack pointer.

– Jumps to a startup routine that copies the program's arguments from the stack to registers and calls the program's main routine.

GLOBAL EDGE
Intelligence Of Things

THANK YOU

GLOBAL EDGE