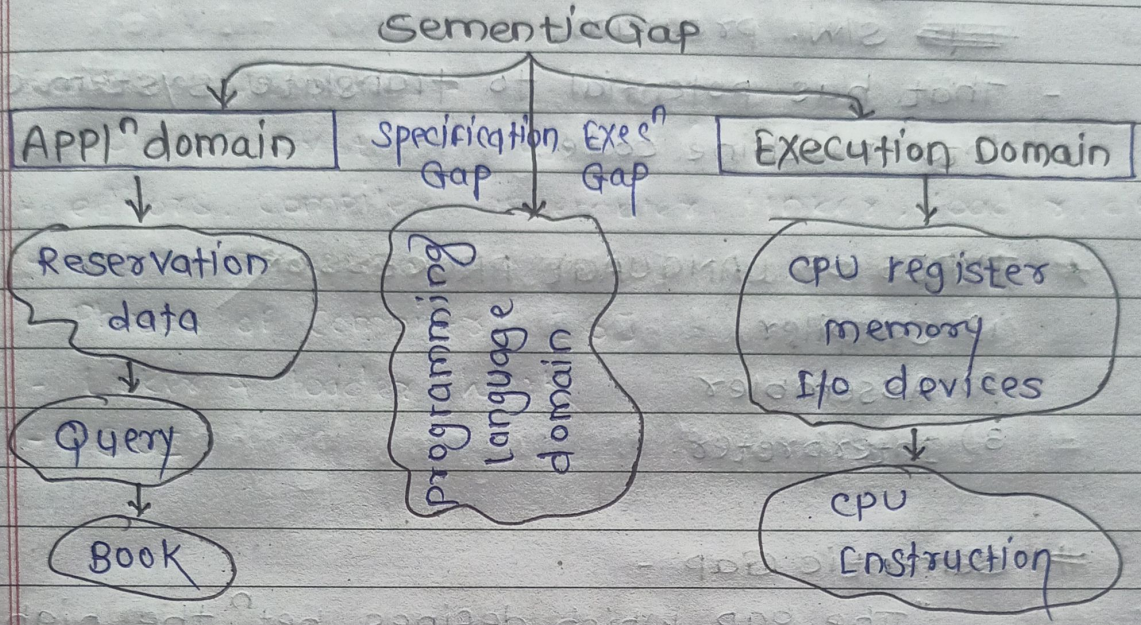
12|Sep|2022
monday

TOPIC-I

Language processors

Language processor :-

"Bridges the gap betⁿ applⁿ domain & Execution domain. by using 'programming language domain'."



System programming :-

- system programming is the activity of programming computer system S/W.
- system programming involves designing & writing of computer programs.
- That allows the computer hardware, interface with the programmer and user.

Application programming -

Application programming provides the service to the particular application only.

for ex. Gaming, WP, Insta.

Language processor :-

- A language processor is the one type of ~~sys tem~~ s/w. program.
- That has potential to translate system code into machine code.

★ TYPES OF LANGUAGE PROCESSOR -

1) Compiler
2) Assembler
3) Interpreter.

Sementic Gap -

"The gap which defines betⁿ the high-level programming set and low level progr--amming set called "sementic Gap".

Applⁿ Domain :- [Requirement analysis, planning.]

Applⁿ domain have isolation properties similar to operating system processes.

- multiple threds can exists within single applⁿ domain.
- An Applⁿ domain is a mechanism used within the common language Infrastructure to isolate execut--ed s/w applⁿ from one other so that they do not affect each other.

14/sep/2022
WED

- Introduction
- Language processor activites
- Fundamental of language processing
- Fundamental of language Specification
- language processing Development tools - LEX & YACC

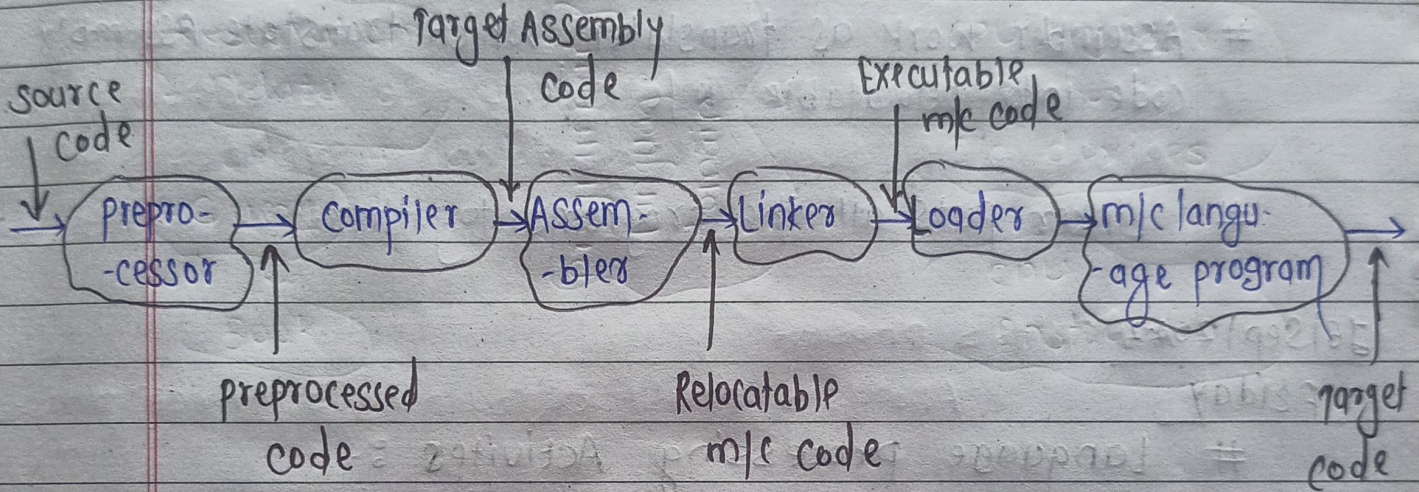


fig:- practical arrangment of language processor.

Preprocesser work for creating error free program.

Linker links all modules.

Lodar occupys the memory space. OR memory allocation

—*——*——

Preprocessor provid Super Set of programing language ~~feta~~ feature.

Compiler Compiles or translate Source Code into machine executable Code.

TESTING TYPES
* Manual Testing - Report
* Automation - No. of Tools are use.

SIX PHASE OF COMPILER
1) Laxical Analyzer - Symbol table generator
2) Syntax Analyzer - error finding
3) Symentic Analyzer - Parssing [grammer use] Tree
4) Intermediate code generator - IC code
5) code optimiser. - ~~कमीत कमी~~ line ~~मध्ये~~ code (Reduce)
6) Target m/c code generator.

Assembler work as translater which translate Assembly code into relocatable m/c code.

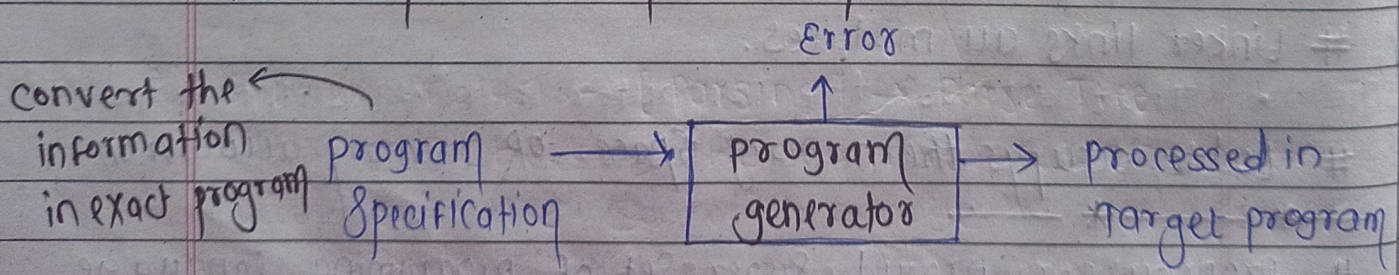
Linker puts all the programs togather.

16|Sep|2022
Friday

Language processing Activites :-

a) program Genezation [Error free program]
b) program Execution
   i) program Translation
   ii) program Interpretation

a) program Genezation Activity :-

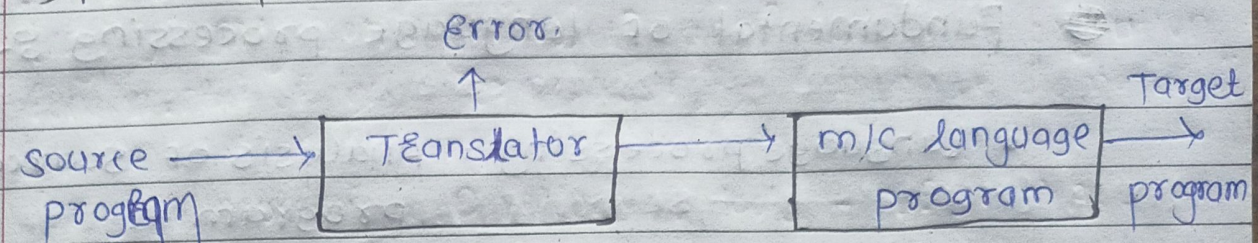


"program generator is s/w system which accept Specifications of program for generation & generates the program in Target program."

ML - machine language

b) program Execution :-

i) program Translator :-



Charachteristics :-

1) program must be translat befor execution.
2) Translated program may save in file, we can execute save file or program repetadly.

ii) program interpreter :-

Three main tasks
1) Fetching instruction
2) Decoding —|— [convert the code in ML]
3) executing —|—

- Interpreter reds the program & store in its memory.

Fundamental of language processing :-

1) Analysis of source program
2) Synthesis of target program

Analysis of source program → (symbol table) creation of Tokans → 1) Lexical analysis
Checking the Format of Sta. → 2) Syntax analysis
Checking the Sta. is meaning or not → 3) symentic analysis

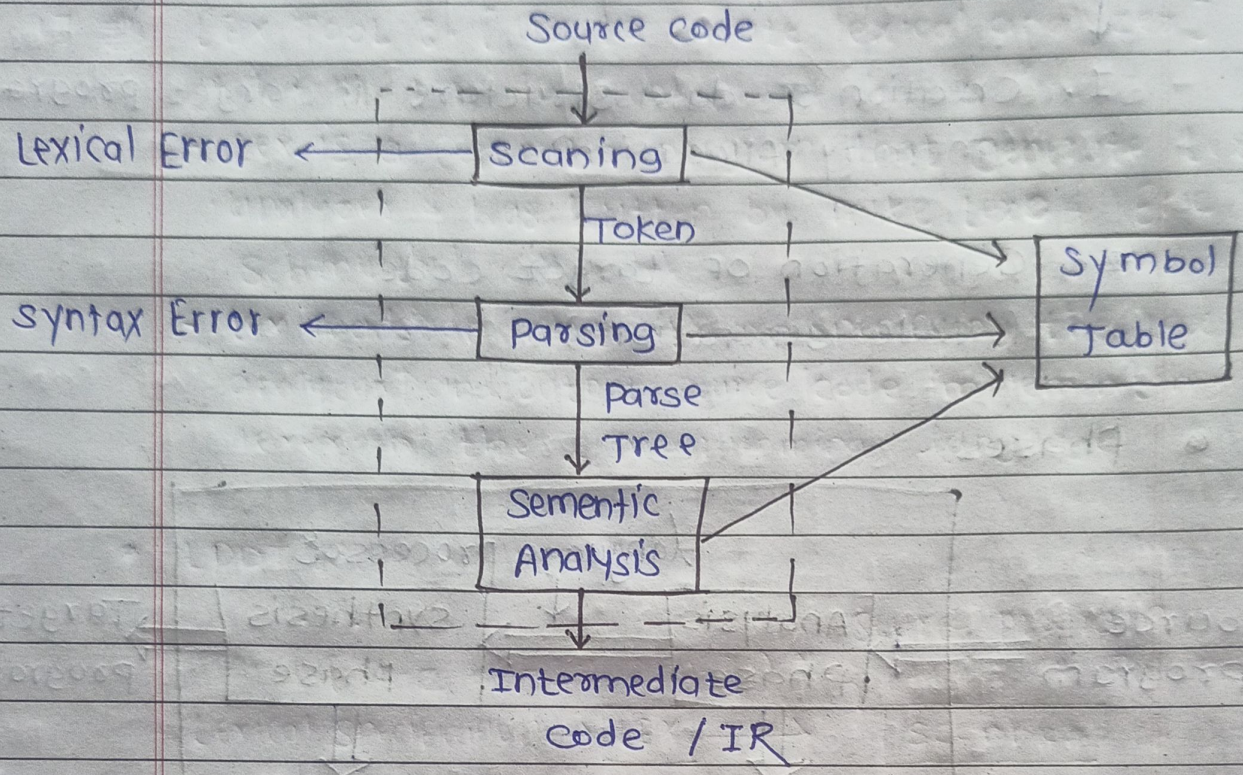
Synthesis of target program → memory allocation & Code generation is Done

For ex - Percent_Profit = (profit * 100) / cost peize

20|sep|2022
Tuesday

IR - Intermediate Representation.

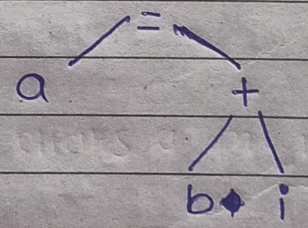
Frount End Toy Compiler :- [Analysis phase]



Example :-

```
integer i;
real a,b;
a = b+i;
```

⟹



Symbol table :-

No.	symbol	type	length	address
1	i	int		
2	a	~~real~~		
3	b	~~real~~		
4	j*	real		
5	temp	real		

Loader will creat .exe file :

- Intermediate code - [Synthesis phase]

1. Convert (id #1) to real giving (id, #4)
2. Add (id, #4) to (id, #3) giving (id, #5)
3. Store (id, #5) in (id, #2).

Back end toy Compiler :- [Synthesis phase]

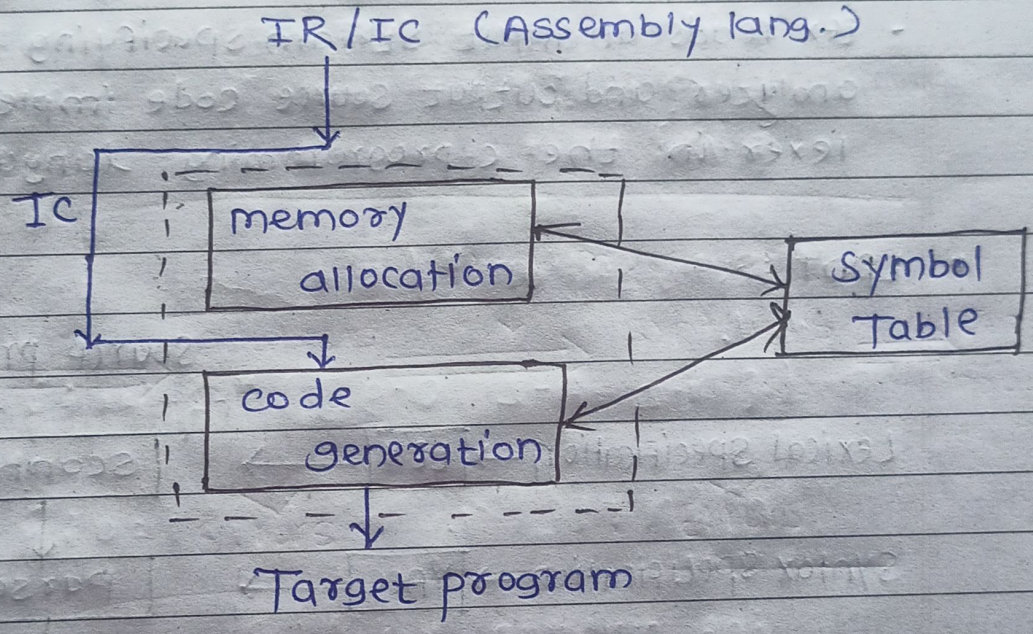


Fig. Back end toy Compiler.

Symbol table - maintain all information about code.

No.	Symbol	TYPE	Length	Address
1.	i	int		2000
2.	a	real		2001
3.	b	real		2002

21/Sep/2022 (Absent)

Wedensday

Flex tool = LEX

LPDT [Language Processor Development Tools].

1] LEX TOOL :- [use to create lexical analyzer]

- Lex is a computer program that generate Lexical analyzer & was written by mike Lexo & Eric Schmidt.
- Lex reads an input stream specifying the lex analyzer and output source code implementing the lexer in the c programming language.

• LPDT :-

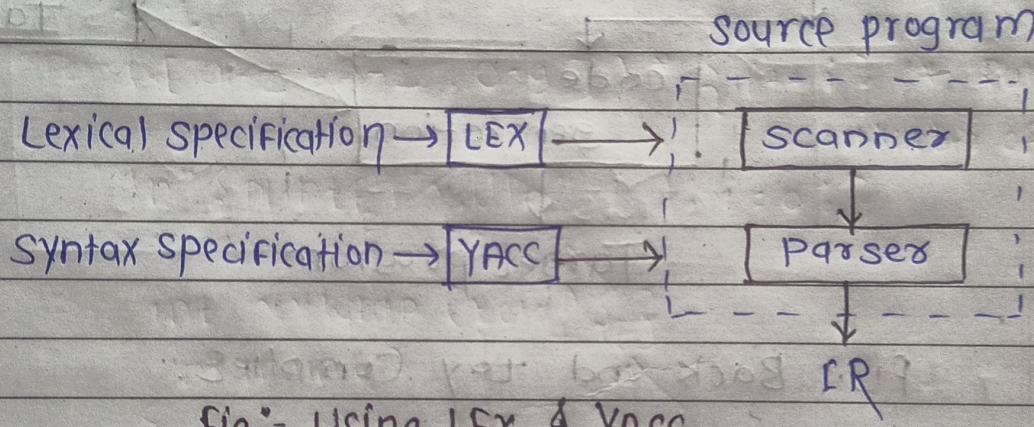
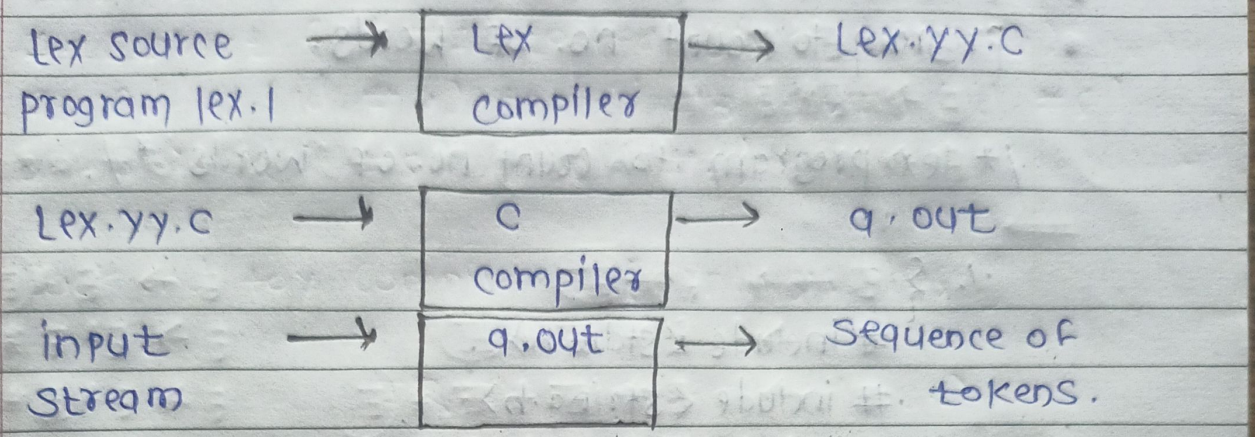


Fig:- Using LEX & YACC

Function of LEX :-

- Firstly lexical analyzer creates a program lex.l the lex language.
- Then lex compiler runs the lex.l program & produce c program lex.yy.c.
- Finally c compiler runs the lex.yy.c program & produce an object program a.out.
- a.out is lexical that transform an input stream into a sequence of tokens.

Source program & memory location are maintain in symbol table [Storing data]



- Content of lex program :-

1) Declarations
2) Translation rules
3) Auxiliary functions.

1) Declaration :- Can Contain declaration of variable & regular definition. & the declaration section can be empty.

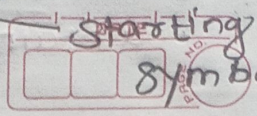
2) Translation rule :- Each of the form pattern followed by action. Each pattern is regular expression. Each action is fragment of C-code.

3) Auxiliary function :-
Starting with the second %%. is optional. Everithing in this section is copied directly to the file lex.yy.c & use in action of translation.

%% %%
divide seperate

Grammer (V T P S)

V → variable
T → Terminal
P → production rule
S → Starting symbol.



- Program to count no. of words.

```
/* lex program to count no. of words */.

%{
  #include <stdio.h>
  #include <string.h>
    int i=0;
    %}

/* Rules Section */
%%
([a-z, A-Z, 0-9]) * {i++ i} /* Rule for
                              Number of word*
"\n" {printf ("y.d\n", i ); i=0; }
%%
int yywrap (void) { }
 int main ( )
{
  // The Fy
```

output :-

```
Lex words.l
cc lex.yy.c -lfl
./a.out
Vishal Dange
  2
This is girl
  3
```

} Commands.

28|Sep|2022
WED

grammer is a set of rule. which is use for define

YAAC (LPOT) Tool :- Yet another Compiler Compiler

- Develop by Stephan C Johan.

Pars tree analysis from left to right.

Grammer :-

"Grammer is set of rule that define valid structure of language."

Gramer is denoted as G = { V, T, P, S }

where,

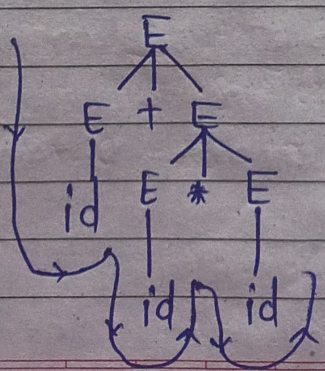
V = Veriable OR nonterminal
T = Terminal symbol
P = Production rule
S = Starting symbol

Example :-

E → E+E / E*E / id Then generate id + id * id

⇒

E → E+E
E → E+ E*E
E → E+ E* id
E → E+ id * id
E → id* id * id



analysis from left to right

2] $E \rightarrow E+E \mid E*E \mid id$

$E \rightarrow E+E$

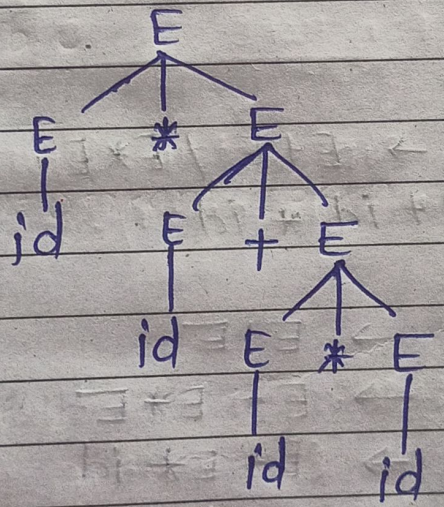
$E \rightarrow E*E$

$E \rightarrow id$ Then generates $id*id+id*id$

$\Longrightarrow$

$E \longrightarrow E*(E)$	$E \rightarrow E+(E)$
$E \longrightarrow E*E+(E)$	$E \rightarrow E+E*(E)$
$E \rightarrow E*E+E*(E)$	$E \rightarrow E+(E)*id$
$E \rightarrow E*E+(E)*id$	$E \rightarrow (E)+id*id$
$E \rightarrow E*(E)+id*id$	$E \rightarrow E*(E)+id*id$
$E \rightarrow (E)*id+id*id$	$E \rightarrow E*id+id*id$
$E \rightarrow id*id+id*id$	$E \rightarrow id*id+id*id$

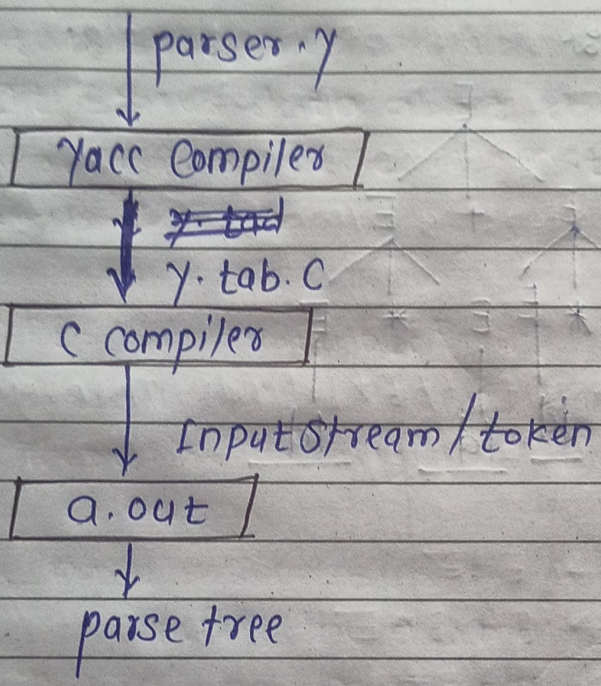
Parse Tree -



Structure of YACC Compiler

Declaration
%. %.
Rules
%. %.
Subrutin & 'C' function.

YAAC Tool :-



- Firstly program will Save as .y extention.
- Then yacc Compiler Compile parser.y & produce y.tab.c C language program.
- The C compiler compile y.tab.c & produce a.out object file.
- Finally a.out obj. file take input as input stream / or token & produce parse tree.

30|Sep|2022
Friday

2. Assemblers

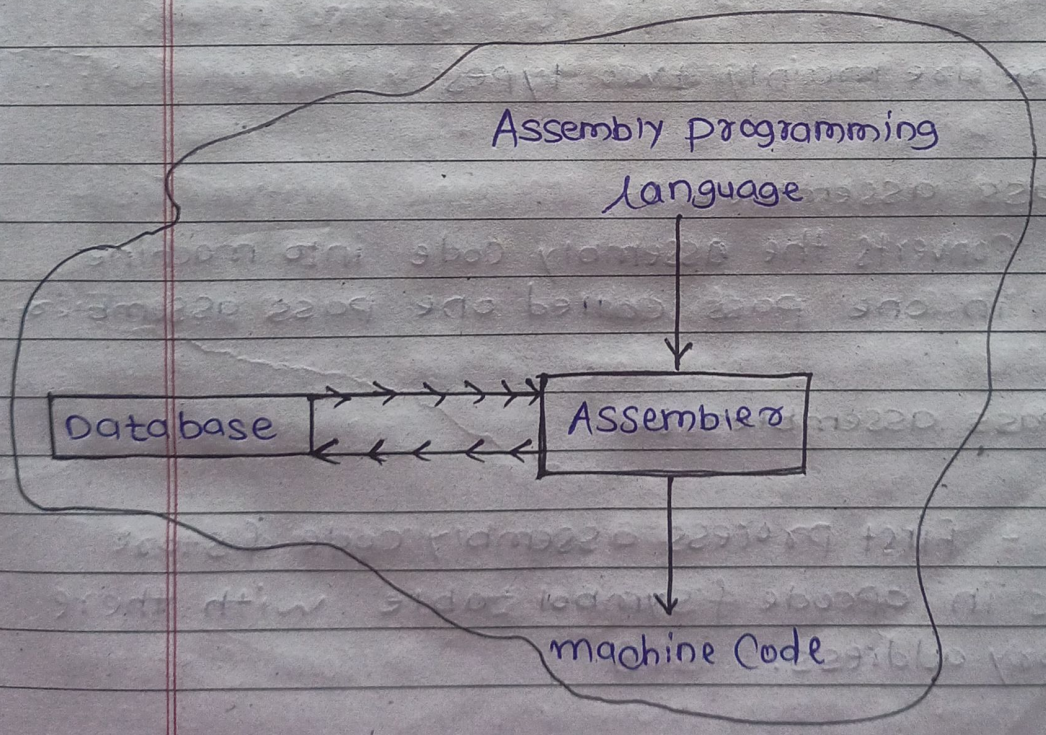
↳ Low level programming S/W OR language.

- ✓ Introduction
- ✓ Element of Assembly programming language
- A simply Asembly scheme [mnemonic]
- Pass structure of Assemblers
  - a] one pass Assembler – Whole code compile.
  - b] Two pass Assembler [Opcode table & Symbol, Target m/c

Introduction :-

"Assembler is a S/W that convert a assembly language code into machine code"

"Symbolic Representation of any instruction is known as mnemonics."



★ Assembly language is low level programming language & It gives inst" to processor for performing different tasks.

Symbolic representation of Instructio
↑ or opcode

opcode & mnemonics :-

↓

Specific information of instruction will takes.

For ex.

ADD A,B

Where,

ADD - is the mnemonics

& A,B are operands

- Database :-

"It stores data about mnemonics with there binary code & instruction called database."

TYPES OF ASSEMBLER :-

There are mainly two types.

1] one pass assembler :-

Converts the assembly code into machine code in one pass called one pass assembler.

2] Two pass assembler :-

pass 1 - First process assembly code & store values in opcode & symbol table with there memory address.

Pass 2 - Finally convert this opcode & symbol table into a target machine code.

07/oct/2022
Friday

Use to create Assembly program

Element of Assembly programing language :-

1) Format of Assembly program Statement
2) Simple set of Ins^n.
3) m/c Instruction format
4) Simple assembly program
5) TYPE of Assembly language statement.

~~format of Assembly program Stat~~ :-

a) menemonic operation code - symbolic representation.
b) Symbolic operand - represend data OR Inst^n
c) Data Declaration - number system

→ operation code for machine inst^n & use numerical operation code & ~~easy~~ easy to remember.

→ programer can Associate symbolic name with data & inst^n & use symbolic name as oprand in assembly stat.

→ Data can declared varietys of notation ~~facet~~ including decimal notation.

Registers [A, B, C, D]

1) Format of Assembly programming statement :-

[Label] <opcode> <operand Specification> [comment]

Where,
- label & comments are optional.
- opcode - mnemonic operational code
- operand specification - first operand is always Reg. & second is memory word with symbolic name.

Example :-

ADD AREG ONE

⇒ Where ADD is the mnemonics
AREG - is the first operand
ONE - is the second operand means memory word with name "ONE".

2) Simple set of Instruction :- mnemonic operational code.

Instruction opcode	Assemby mnemonic	Remark
00	STOP	Stop execution
01	ADD	perform addition
02	SUB	perform Substraction
03	MUL	perform multiplication
04	MOVER	move from memory to register

05	MOVEM	move from Reg to memory
06	COMP	Compare & set condition code
07	BC	Branch on Condition
08	DIV	Perform division
09	READ	Read into register
10	PRINT	print Content of reg.

* BC - [Branch on Condition] Conditional Code :-

Format :→ BC <condition code specification>, <memory Address>

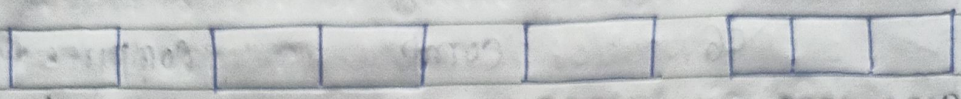
↓

LT - Lower than
LE - Lower than Equal to
EQ - Equal TO
GT - Grater Than
GE - Greater than equal to
ANY - Unconditional control transfer.

10/Oct/2022
Monday

3] Machine Instruction Format :-

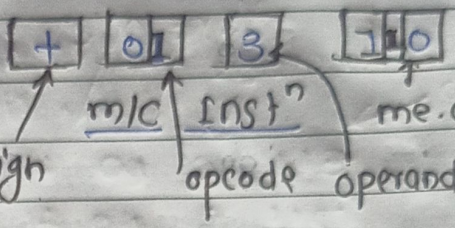
Format -



For. ex.

ADD CREG , ONE

Assembly Inst^n



- The opcode, reg. operand, memory add^r, occupies two, one & three degits respectively.

4] TYPES of Assembly language statement :-

There are three types -

Program execution ← A) Imperative statement - Assembly Inst^n

B) Declarative statement

C) Assembly Directives.

A] Imparative statement :-

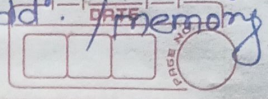
- The statement perform action during assembly / program execution called Imparative statement
- Assembly language Inst^n

EX.

MOVER CREG, BREG.

Content of BREG move into the CREG. Using the movR Inst^n called imparative inst^n. / statement

Constant value always represent add^{n}. of memory blocks



B) Declarative Statement :-

i) Declarative storage

ii) Declarative Constant

i) Declarative storage :-

Syntax - [label] DS <constant> ← memory blocks

It reserves area of memory & associate symbolic name.

ex. ABC DS 200

ii) Declarative Constant -

Syntax - [Label] DC '<value>' ← Const. Value with mem. add^{r}.

It construct memory word OR memory add^{r}. Containing constant

ex. XYZ DC '1'

c) Assembly Directives :- It can not generat m/c.

i) START - They are use to instruct

ii) END. Assembler for action / inst^{n}.

i) START :-

- program start with START <200> inst^{n}.

syntax :- ex.

START <constant>

- It indicates that first word of target program which start from Rom memory location with add^{r}.

iii] END -

- It indicates END of Source program.

Syntax -

END.

5] Simple Assembly program :-

		Location Counter	sign	opcode	reg. operand	addr
START	101					
READ	N	101	+	09	0	113
MOVER	BREG, ONE	102	+	04	2	115
MOVEM	BREG, TERM	103	+	05	2	116
MUL	BREG, TERM	104	+	03	2	116
MOVER	CREG, TERM	105	+	04	3	116
ADD	CREG, ONE	106	+	01	3	115
MOVEM	CREG, TERM	107	+	05	3	116
Comp	CREG, N	108	+	06	3	113
BC	LE, AGAIN	109	+	07	2	104
MOVEM	BREG, RESULT	110	+	05	2	114
PRINT	RESULT	111	+	10	0	114
STOP		112	+	00	0	000
DS	1	113				
DS	1	114				
DC	'1'	115	+	00	0	001
DS	1	116				
END						

12/oct/2022
WED

5 Data structure present in assembler

DATE
PAGE NO.

A simple assembly Scheme :-

1] Design Specification for an assembler.
2] Phases of assembler
3] Data structure used by assembler. [storing & Featching Data]

1] Design Specification for an Assembler :-

a] Specify problem & identify the info.
b] Specify Data structure
c] Specify format of data structure
d] Specify algorithm use to maintain the information.

2] Phases of Assembler :-

- a] Analysis phase - Work on data structure.
- b] synthesis phase - synthesis the info. in symbol table.

Symbol table is generate & mnemonics table will also generate with there memory address.
- It analysis the source program & work on different Data structure. And actual program is scanned.

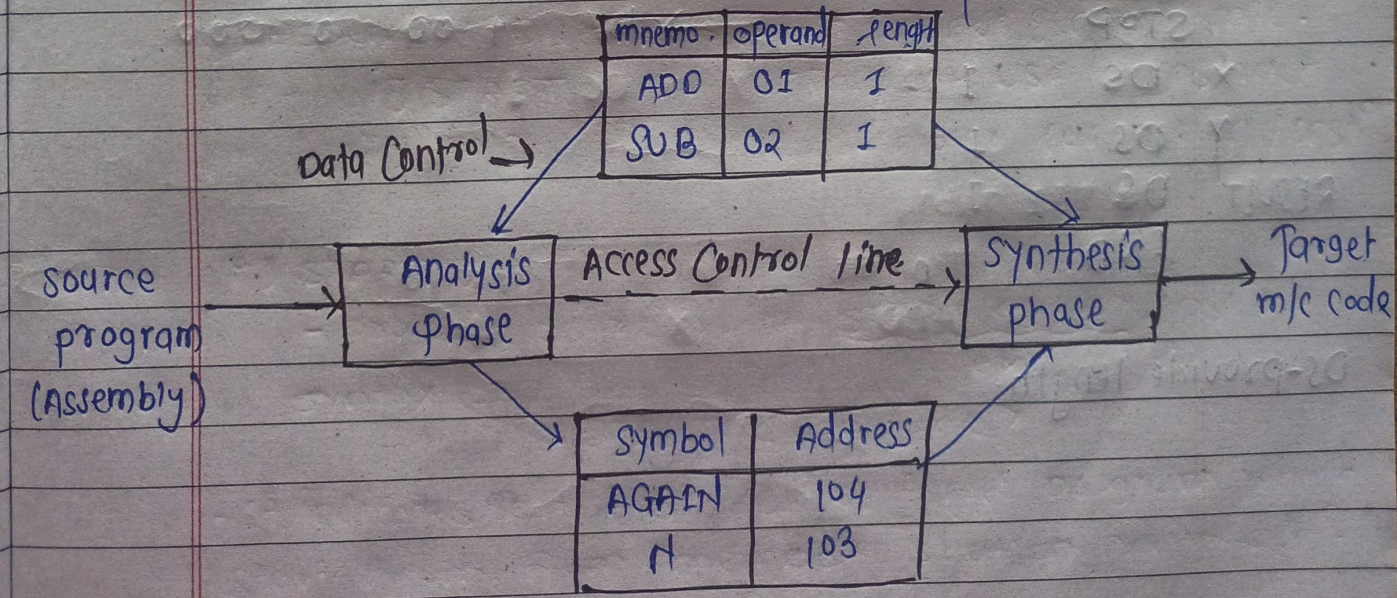


Fig :- TWO-Pass Assembler.

a] Analysis phase :-
- symbol table building
- memory allocation
- Location Counter [info. about next Inst^{n} I- Add^{r}

b] Synthesis phase.
- obtain opcode from memonic table.
- obtain memory add^{r} from symbol table
- synthesis machine Inst^{n}.

Addition program :-

			LC		m/c	Inst^{n}
	START	101				
	READ	X	→ 101	+ 09	0	108
	READ	Y	→ 102	+ 09	0	109
	MOVER	AREG, X	→ 103	+ 04	1	108
	ADD	AREG, Y	→ 104	+ 01	1	109
	MOVEM	AREG, RESULT	→ 105	+ 05	1	112
	PRINT	RESULT	→ 106	+ 10	0	112
	STOP		→ 107	+ 00	0	000
X	DS	1				
Y	DS	1				
RESULT	DS	1				
	END					

DS-provide length.

3] Data structure used by Assembler :-

a] Symbol Table - (SYMTAB)
b] Literal Table - (LITTAB)
c] Pseudo_opcode Table - (OPTAB)
d] Location Counter - - (LC) contain add' of next Inst"
e] pool Table - (POOLTAB) moment, pointer add'
f] mnemonic Table - (MOT)

a] Symbol Table -

Index No	Symbol	Add'	length
1	loop	202	1

b) Literal Table -

Index No	literal	Add'
1	= '5'	211
2	= '1'	212

c] Pseudo opcode Table -

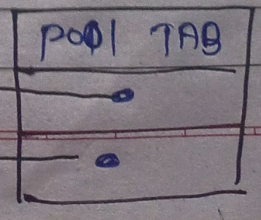
mnemonic opcode	class	mnemonic info
START	AD	R#1

d] Location Counter -

Variable	Add'
X	108
Y	109

e] Pool Table.-

literal	Add'
= '5'	100
= '1'	101



17/oct/2022
Monday

DC - Declarative constant

TII - Table of Incomplete inst^n.

Pass structure of Assembler :

1] Single pass Assembler
2] Two Pass Assembler

Single pass Assembler -

- Assembler ~~scan~~ scan input file in once
- Generate target machine code
- It is a faster than multipass Assembler.
- It contain symbol, litteral, mnemonic & opcod table as data structure.
- In that forward reference problem arrises it means a veriable is found before define. To solve this forword reference problem use Backpatchines as a TII table [Forward reference Table]

Example :-

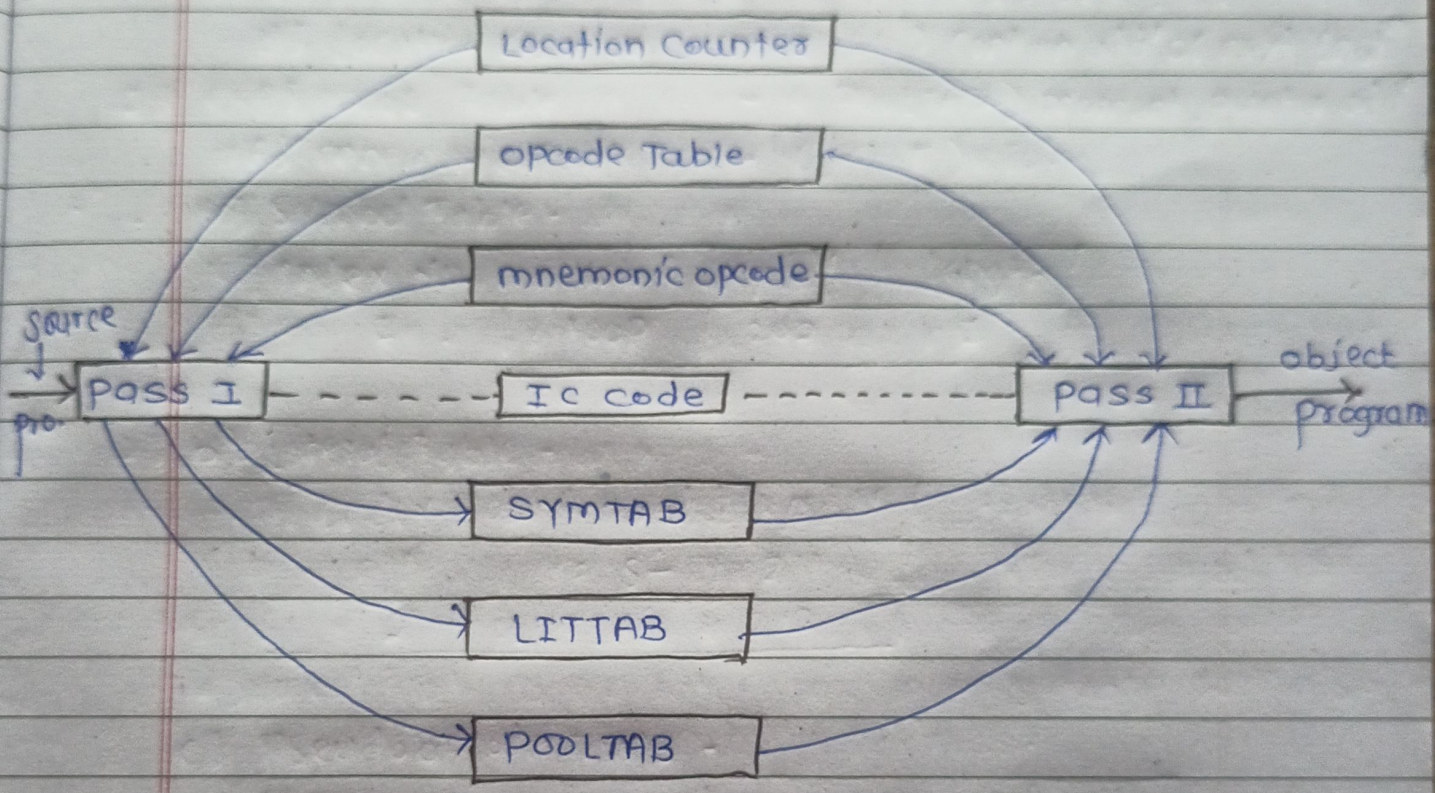
Line No.			
4	ADD	R1, X	
22	X	DC	1

TII Table

Type	Value	Line No
ADD	X	4

TWO pass / multipass Assembler :-

I] pass 1 - [Analysis]
II] pass 2 - [Synthesis phase]



I] pass-I :-
- Seperate different data structure. OPTable, MOT etc.
- Built SYMTAB & LITTAB.
- perform Location Counter processing
- Construct intemediate code.

II] pass-II :-
- Synthesis of target program.
- Execute fild & generate a mc.

Mathematics

Question 1

Which of the following describes the end behavior of f(x) = 2x / 3x^2-3 ?

1. The graph approaches 0 as x approaches infinity.

2. The graph approaches 0 as x approaches negative infinity

3. The graph approaches 2/3 as x approaches infinity.

4. The graph approaches -1 as x approaches negative infinity.

Question: - 2.

Let f(x) =4-x^2, g(x) =2-x find (f+g) (x) and its domain

Chemistry

Question: - 1.

where are the electrons found in Bohrs atomic atom

A. the electrons orbit the protons at the atoms center.

B. the electrons circle the nucleus in specific orbits.

C. the electrons occupy the atoms center, with protons orbiting.

D. the electrons are evenly distributed throughout the atom.

Question: - 2.

A molecule has two atoms that are bonded to the central atom and one lone pair of electrons around this central atom. What is the shape of the molecule? Bent linear trigonal planar trigonal pyramidal

Biology

Question: - 1.

Lactic acid fermentation is a type of anaerobic respiration that occurs in organisms such as:

A. Bacterial Cells

B. Animals

C. Yeast Cells (fungi)

D. Plants

Question: - 2.

The chemical and mechanical processes of food breakdown are called _____.

A. digestion

B. absorption

C. ingestion

D. secretion

Physics

Question: - 1.

Boyle's law relates which of the following gas properties?

a) pressure and temperature

b) Pressure and volume

c) Temperature and density

Question: - 2.

Explain how atomic mass and molecular mass are determined

Health

Question: - 1.

What are examples of environmental lung diseases? Check all that apply. pneumoconiosis bronchitis cystic fibrosis silicosis byssinosis anthracosis asbestosis tuberculosis

Question: - 2.

When an individual feels unable to deal with the situation which of the following may be used

History

Question: - 1.

label the social hierarchy of feudalism Label A Label B Label C Label D Nobles King Peasants Knights

Question: - 2.

Which phrase describes a tactic that a lobbyist would use to influence public policy

Social Studies

Question: - 1.

Imagine that your best friend tells you that he doesn't think the water cycle is important. What would you tell him?

Question: - 2.

Please place each step in the correct order: citizens vote, electors vote,electors choose the president, electors are chosen.

Geography

Question: - 1.

As a religious faith, candomblé appropriates beliefs from both _____.

A. African faiths and catholicism

B. Hinduism and catholicism

C. Hinduism and islam

D. . Buddhism and catholicism please select the best answer from the choices provided. .

Question: - 2.

Compare and contrast the independent and dependent variables in an experiment. give an example of each.

Arts

Question: - 1.

In which way is music used in the 2014 Nike advertisement "Risk Everything," starring popular football champions?

A. lyrical language

B. authority establishment

C. memorability

D. targeting

E. continuity

Question: - 2.

Answer the following question in 3-4 complete sentences. An elaborately illustrated page in a Medieval Bible. What is the above image an example of? Describe its purpose and importance in history.

English

Question 1

How do Scout and Jem respond differently to this new information about Bob Ewell? Which child

most likely has a more accurate assessment of what will happen in light of this new evidence? Why?

Question 2

Choose the sentence in which the word

ball is a direct object.

A. The pitcher threw the ball.
B. The ball bounced on the ground.
C. That muddy blob is the ball.
D. The batter swung above the ball.

Buisness

Q1.

an advantage of a partnership compared to a sole proprietorship is that a partnership multiple choice question. is simpler than other forms of organization. is cheaper than other forms of organization. is less complex than other forms of organization. is able to raise larger amounts of capital.

Q2.

dan picked up his friend rodney to drive to their softball game. both dan and rodney have a personal auto policy (pap) with $5,000 of medical payments coverage. dan hit a parked car, and rodney was injured, incurring $9,000 of medical expenses. how will this claim be settled under the other insurance provision of the pap?