Topic - 1

13/sep/2022

Classical Encryption Techniques & DES
[Data Encryption Standard]

[a] OSI Security architecture -

1] Security Attack
2] security mechanism
3] Security services

Q. What is threats?

⇒

"A threat is a possible security risk that might exploit the vulnerability of system."

Hacking information OR the data from other way there will no any security.

Q. What is Attack?

⇒

"An attack is an intentional unauthorized action on system."

Directly attack on system and break the security & Hack the information called attack"

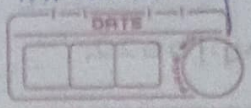
1] Active attack -

An active attack is an attempt to change the system resource.

[Modified the information]

14/Sep/2022
WED

goal of passive attack is obtain information that is being transmited

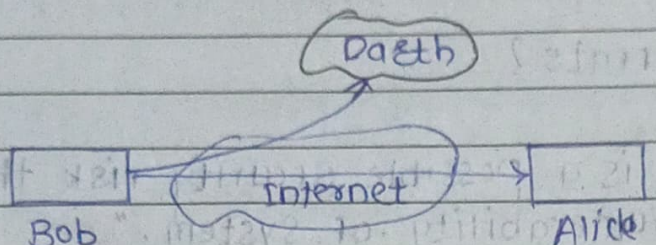


1] Passive attack. [only listening]

A passive attack is an attempt to understand or retrive data from system without influencing. OR modification

TYPES of Passive Attack :-

1] Release of message content



2] Traffic Analysis - Encryption pattern will be obtain by third party. and understanding the communication bet^n the sender & reciver.
[and exchange the message or data bet^n sender & reciver

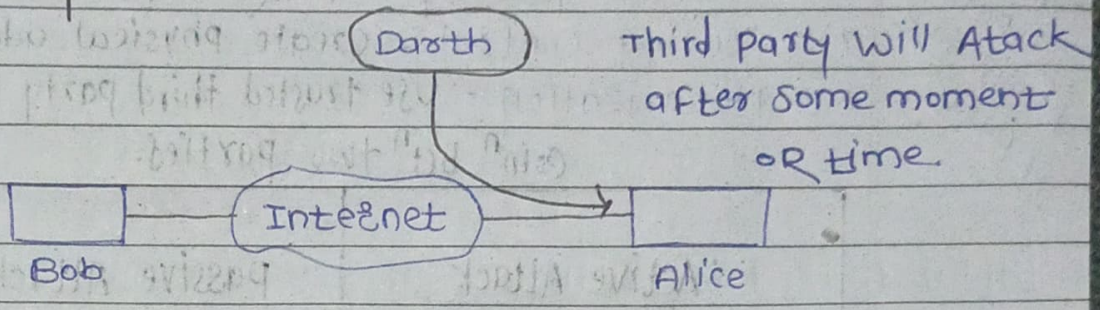
1] Release of message content -

- When the any message or data will send by the sender & reciver get this data.
- Hence, this process will carried out again & again but both sender & reciver is no idea about his data OR convertiation will read by the third party after he will not disturb to sender & reciver.
- third party OR Hacker will only learn the conversiation.

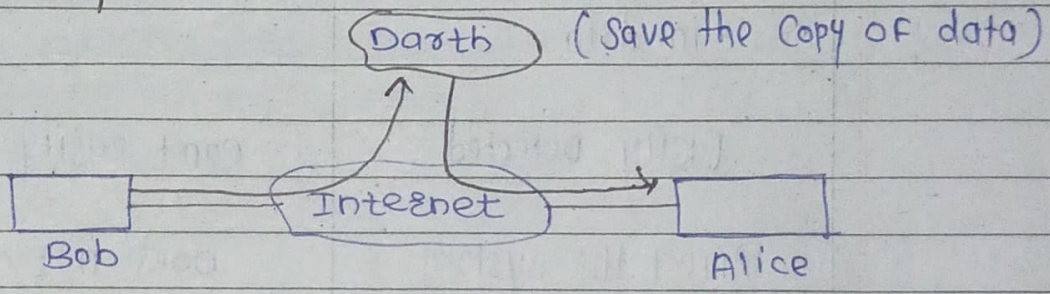
2] Active Attack :- modification of data.

1) masqureds - Third party will send the message.
2) Replay - message will change by the third party
3) Modification of message -
4) Denial of Service (DoS)

1] Masqueeads :-



2] Replay :-



3] Modification of message -
modification will be done by the third party and send the data for the Eeciverѕ.

4] Denial of Service (DoS) :-
Third party will creates the number of IP address so bob is not getting the service from the Server side. Becaus Server is overloded by the third party.

* A process that is designed for detect, prevent & detect the ~~recover from~~ attack.

Security Mechanism :-

1) Encipherment - Hiding or Covering data, Planetext to ciphertext
2) Digital Signature - append short check value of data.
3) Access Control - Emitations are require like ID & Pass.
4) Data integrity - sender can electronically sign data & recver can electronically verify the
5) Authentication Exchange - Information exchange.
6) Traffic padding - Fack Frames added in datastream
7) Routing Control - Create physical address.
8) Notarization - use trusted third party to control com^{n} bet^{n} two parties.

[code is know only sender & reciver]

No modification of data & No miss use of data

Active Attack	Passive Attack
Modification will done in data	No modification
Easily detected	Can't easily detect
Affect the system	Does not affect to system
~~Easy~~ ~~to~~ Difficult to prevent	Easy to prevent
TYPES :- - masquerads - Replay - modification of message	TYPES :- - Release of the message - traffic analysis

16/Sep/2022
Friday

2] # Security Mechanism :-

- Types of Security mechanism :-

1) Specific security mechanism :-
(Related to the protocol)

2) Pasivasive security mechanism :-
↓ (Different polysis)

a) Trusted Functionality - Specified policies
b) Security label - For Identification the resource.
Security purpose → c) Event detection - For detecting the attack
d) Security Audit Trail - We hace to check the Collection is Correct OR not
e) Security recovery.
↓
most important thing because recover for the attack & get the data into original position.

3] # Security Services :-

1) Authentication (protection) - a] Peer entity - Root of message is fixed.
b] Data origin - sended data is original
2) Access Control - Limitation on internet
3) Data Confidentiality - maintain the data with us only
4) Data integrity - No any modification OR file will not delete
5) Nonrepudiation - the things we are Sending to Reciver then Reciver not denail the message will geating to me.

19/sep/2022
monday

Cryptography :-

"It is a technique in which plain text is converted into the cipher text using encryption & decryption keys called cryptography and again cipher text is convert in plain text

Plain Text → Encryption Algorithm → Cipher Text → Decryption Algorithm → Plain Text

1) Public key
2) private key

- symetric cryptography :- "For the purpose of encryption ~~purpose~~ & decryption ~~purpose~~ same key will be provided is called as symmetric cryptography" and also known as private key cryptography.

- Asymmetric key cryptography :- "For the purpose of encryption public key is use & for Decryption private key is used called Asymmetric key cryptography."

OR

"public & private keys are uses for Decryption & encryption called Asymmetric key cryptography."

NOT In Syllabus :-

- Encryption Scheme :-
  1) Unconditional Scheme
  2) Computational Scheme.

★ Symmetric cipher :-

1) Substitution Technique - Changing the latter byanother
2) Transposition Technique - only change positions.

1) Substitution Techiques -

Latters are replaced by othe letter of symbols.

For ex -

B → x
a → M
g → A

Bag → xMA → Bag

cipher text

2) Transpositon Technique -

Applying some sort of permutati-on on plain text letters.

For ex -

NESo

→ ESoN, SNEO, OSEN, SONE, ENOS

21/sep/2022
wedensday

Substitution :-

In this cipher the text is convert from plain text to cipher text on base of key.

1] Caesar cipher :-
- oldest
- Easy to implement

Algorithm :-

'P' → 'C'

K = 3 Fixed value.

1) $C = E(P_1 k) \bmod 26$
$= (P+k) \bmod 26$
$P = D(C_2 k) \bmod 26$
$= (C-k) \bmod 26$

Cryptography :-

FUBSWRJUDSKB

11] Shift cipher :-

As Compare to Caesar cipher, shift cipher key cannot break, because the key value changes everytime from 0-25. this method takes extra time as Compare to Caesar cipher.

Monoalphabetic cipher :-

The cipher line can be any permutation of 26 Character/ alphabet.

e.g.

$S = \{a, b, C\}$

a = m

every time we will change 'a' as m is generated. To overcome broot force attack this method every time e.g.

G z GE WVGR NCP.

EXECUTE PLAN ← Guising.

27/Sep/2022

Tuesday

Playfair Cipher :- [Digraph method]

1) Create Diagram - 5x5 matrix
2) Repeating letter - filer letter
3) Same column (↓) wrap around
4) Same Row (→) wrap around
5) Rectangle (⇄) Swap

key - Monarchy.

M	O	N	A	R
C	H	Y	B	D
E	F	G	I/J	K
L	P	Q	S	T
U	V	W	X	Z

Plain Text :- attack

at	ta	ck	← PT
RS	SR	DE	← CT

Plain Text :- mosque

mo	sq	ue	← Plain Text
ON	TS	mL	← cipher Text

- Filer letter :- balloon

ba ll oo nx ← Filer letter.

Filer letter fils at that position where repetation take place, after fileing this Filer letter we can stop repetation.

forex :-

balloon

ba ll oo nx

ba lx lo on

V. Imp - 8 marks

Example

- Hide the gold under the carpet.
  Key - plaintext

⇒

P	L	A	I/J	N
T	E	X	B	C
D	F	G	H	K
M	O	Q	R	S
U	V	W	Y	Z

hide the gold under the carpet (X) ← Filler letter

hi	de	th	eg	ol	du	nd	er	th	ec
RB	FX	BD	XF	VE	MP	PK	BO	BO	XT

ar	pe	tx
JQ	LT	EB

A	I/J
X	B
Q	H
Q	R

X is the filler letter

ciphertext is →

RB FX BD XF VE MP PK BO BO XT JQ LT E

Plain Text & matrix of 3x3 will takes

↑

Hill Cipher :- [multiletter cipher]

- multiletter cipher
- "Lester Hill" in 1929 was developed
- Encrypt a group of letters
  - diagraph, trigraph, polygraph

Algo. :-

1) $E = E(K, P) = PX \; K \bmod 26$
2) $P = E(K, C) = CX \; K^{-1} \bmod 26$
   $= PXK \times K^{-1} \bmod 26$
   $= P \bmod 26$

Example :-

PlainText = Pay more money

$$K = \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix} \quad \cdot \begin{bmatrix} P & a & y \\ 15 & 0 & 24 \end{bmatrix}$$

$$\text{cipher Text} = \begin{bmatrix} 15 & 0 & 24 \end{bmatrix} \times \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

$$= \begin{bmatrix} 15X17 + 0X21 + 24X2 \\ 15X17 + 0X18 + 24X2 \\ 15X5 + 0X21 + 24X19 \end{bmatrix}$$

$$= \begin{bmatrix} 303 & 303 & 531 \end{bmatrix} \bmod 26$$

$$= \begin{bmatrix} 17 & 17 & 11 \end{bmatrix}$$

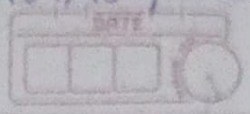
R R L

plain text = pay

Cipher text = RRL

A b c d e f g h i j k l m n o p q r s t u v w
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22

x y z
23 24 25



- [m o ε]
  12 14 17

$$\text{cipher Text} = [12 \;\; 14 \;\; 17] \times \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

$$= \begin{bmatrix} 12\times17 + 14\times21 + 17\times2 \\ 12\times17 + 14\times18 + 17\times2 \\ 12\times5 + 14\times21 + 17\times19 \end{bmatrix}$$

$$= \begin{bmatrix} 204 + 294 + 34 \\ 204 + 252 + 84 \\ 60 + 294 + 323 \end{bmatrix}$$

$$= [532 \;\; 490 \;\; 677] \bmod 26$$

$$= [12 \;\; 22 \;\; 1]$$

↓

= M W B

plain text = moε
cipher text = MWB

- [e m o]
  4 12 14

$$\text{cipher text} = [4 \;\; 12 \;\; 14] \times \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

$$= \begin{bmatrix} 68 + 252 + 28 \\ 68 + 216 + 28 \\ 20 + 252 + 266 \end{bmatrix}$$

$= [348 \quad 312 \quad 538] \bmod 26$

$= [10 \quad 0 \quad 18]$

↓

= K A S

Plain text = emo

cipher text = KAS

- [n e y]
  13 4 24

$$\text{cipher text} = [13 \quad 4 \quad 24] \times \begin{bmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{bmatrix}$$

$$= \begin{bmatrix} 221 + 84 + 48 \\ 221 + 72 + 48 \\ 65 + 84 + 456 \end{bmatrix}$$

$= [353 \quad 341 \quad 605] \bmod 26$

$= [15 \quad 3 \quad 7]$

↓

= P D H

Plain text = n e y

cipher text = P D H

Final Result:-

Plain text = Pay more money

cipher text = RRL MWB KAS PDH

28/Sept/2022

WED

Cipher text to plain text Conversion

- Plain Text :- [usually ordinary readable text]

$$P = D(k, C)$$
$$= C \times K^{-1} \bmod 26$$

$$K^{-1} = \left(\frac{1}{\text{determinant}} \times \text{Adjoint} \right) \bmod 26$$

$$\text{Det} = \begin{pmatrix} 17 & 17 & 5 \\ 21 & 18 & 21 \\ 2 & 2 & 19 \end{pmatrix} \bmod 26$$

For example :-

$$|A| = \begin{vmatrix} +a & -b & +c \\ d & e & f \\ g & h & i \end{vmatrix}$$

$$\det(A) = a * \begin{vmatrix} e & f \\ h & i \end{vmatrix} - b * \begin{vmatrix} d & f \\ g & i \end{vmatrix} + c * \begin{vmatrix} d & e \\ g & h \end{vmatrix}$$

$$= aei - afh - bdi + bfg + cdh - ceg$$

$$\text{Det} = 17 \times \begin{vmatrix} 18 & 21 \\ 2 & 19 \end{vmatrix} - 17 \begin{vmatrix} 21 & 21 \\ 2 & 19 \end{vmatrix} + 5 \begin{vmatrix} 21 & 18 \\ 2 & 2 \end{vmatrix}$$

$$= 17 * (342 - 42) - 17 * (399 - 42) + 5 * (42 - 36)$$

$$= (5100 - 6069 + 30) \bmod 26$$

$$= (-939) \bmod 26$$

$$\text{Det} = 23$$

$$\text{Adj} = \begin{vmatrix} + & - & + \\ 17 & 17 & 5 \\ -21 & +18 & -21 \\ +2 & -2 & +19 \end{vmatrix}$$

$$= \begin{vmatrix} (18\times19-21\times2) & -(21\times19-2\times21) & +(21\times2-18\times2) \\ -(17\times19-2\times5) & +(17\times19-2\times5) & -(17\times2-17\times2) \\ +(17\times21-18\times5) & -(17\times21-21\times5) & +(17\times18-21\times17) \end{vmatrix}$$

$$= \begin{vmatrix} 300 & -357 & 6 \\ -313 & +313 & 0 \\ 267 & -252 & -51 \end{vmatrix} \quad \text{mod } 26$$

Then now Convert column to row

$$\text{Adj} = \begin{vmatrix} 300 & -313 & 267 \\ -357 & 313 & -252 \\ 6 & 0 & -51 \end{vmatrix} \quad \text{mod } 26$$

$$\text{Adj} = \begin{vmatrix} 14 & -1 & 7 \\ -19 & 1 & -18 \\ 6 & 0 & 1 \end{vmatrix} \quad \text{mod } 26$$

$$= \begin{vmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{vmatrix}$$ add 26 at the number.

Then

$$K^{-1} = \left(\frac{1}{D} \times \text{Adj}\right) \text{ mod } 26$$

$$K^{-1} = \frac{1}{23} \begin{vmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{vmatrix} \bmod 26$$

$$= 17 \begin{vmatrix} 14 & 25 & 7 \\ 7 & 1 & 8 \\ 6 & 0 & 1 \end{vmatrix} \bmod 26$$

$$= \begin{vmatrix} 238 & 425 & 119 \\ 119 & 17 & 136 \\ 102 & 0 & 17 \end{vmatrix} \bmod 26$$

$$K^{-1} = \begin{vmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{vmatrix}$$

$$P = C \times K^{-1} \bmod 26$$

- **RRL**

$$(17\ 17\ 11) \times \begin{vmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{vmatrix} \bmod 26$$

$$\begin{vmatrix} 68 + 255 + 264 \\ 153 + 289 + 0 \\ 165 + 102 + 187 \end{vmatrix} \bmod 26$$

$$(587 \quad 442 \quad 544) \bmod 26$$

$$(15 \quad 0 \quad 24)$$

$$RRL = P \quad a \quad y \leftarrow \text{plain text}$$

- MWB

(12 22 1)

$$(12\ 22\ 1) \times \begin{vmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{vmatrix} \bmod 26$$

$$\begin{vmatrix} 48+330+24 \\ 108+374+0 \\ 180+132+17 \end{vmatrix} \bmod 26$$

$$(402 \quad 482 \quad 329) \bmod 26$$

$$(12 \quad 14 \quad 17)$$
$$\downarrow \quad \downarrow \quad \downarrow$$

MWB = m o r

cipher Text — plain text

- KAS

(10 0 18)

$$(10\ 0\ 18) \times \begin{vmatrix} 4 & 9 & 15 \\ 15 & 17 & 6 \\ 24 & 0 & 17 \end{vmatrix} \bmod 26$$

$$\begin{vmatrix} 40+0+432 \\ 90+0+0 \\ 150+0+306 \end{vmatrix} \bmod 26$$

$$(472 \quad 90 \quad 456) \bmod 26$$

$$(4 \quad 12 \quad 14)$$
$$\downarrow \quad \downarrow \quad \downarrow$$

KAS = e m o

(POH)

(15 3 7)

(15 3 7) *	4	9	15	mod 26
	15	17	6	
	24	0	17	

60 + 45 + 168	mod 26
135 + 51 + 0	
225 + 18 + 119	

(273 186 362) mod 26

↓ ↓ ↓

POH = (13 4 24)

↓ ↓ ↓

PDH = n e y ← plain text

Opher text

04/oct/2022
Tuesday

polyalphabetic cipher :- [vignere cipher]

$Ci = (Pi + Ki) \bmod 26$
$Pi = (Ci + Ki) \bmod 26$

key value will be repeat at the end of plain text

ex. key - deceptive
PT - wearediscovered save yourself
CT - ZICVTWQNGRZGVTWAVZHCQYGLMGJ

key	d	e	c	e	p	t	i	v	e	d	e	c	e	p	t	i	v	e	d	e	c			
PT	w	e	a	r	e	d	i	s	c	o	v	e	r	e	d	s	a	v	e	y	o			

key	e	p	t	I	v		e
PT	u	r	s	e	i		f

key	3	4	2	4	15	19	8	21	4	3	4	2	4	15	19	8	21	4	3	4	2	4	15	19	8
PT	22	4	0	17	4	3	8	18	2	14	21	4	17	4	3	18	0	21	4	24	14	20	17	18	4
CT	25	8	2	21	19	22	16	13	6	17	25	6	21	19	22	0	24	25	7	2	16	24	6	11	12

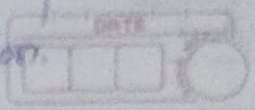
key	21	4
PT	11	5
CT	6	9

Autokey - use to overcome repetation of words.

- In this case we have key & PT. In key we can add PT into key as long PT
- vignere propose autokey system in which keyword is concatinated with PT. itself to provide running key.

→ more secure.
→ unbreakable.

- Apply on binary numbers.
- XOR operation.

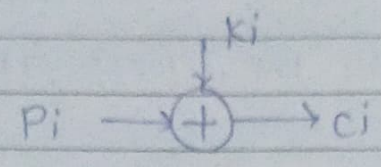


Veenam cipher :- [Gilbart 1918]

$C_i = P_i \oplus k_i$

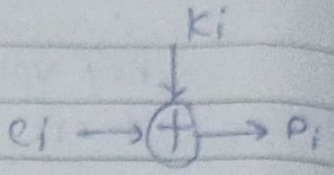
$P_i = C_i \oplus k_i$

Cryptographic Stream generator



Encryption

Cryptographic Stream generator



Decryption.

- method - One time Pad. (only one time key is use)
  - Different keys are use for encryption.

Disadvantage -

- ~~Receiver~~ Reciver dont know about key. which used by sender.

- One time pad :-
  - improvment to verram ~~cipher~~ cipher.
  - It yileds ultimate insecurity
  - Random key is used. as long as message.
  - The key need not be repeted
  - The key which is used to encrypt & decrypt a single msg. and then it is discorded.
  - Each new message requires a new key of same length. as the new msg.
  - Such a schem known as one time pad which is unbreakable.

- Disadvantages :-
  - practical problem of making large quantity of Random key
  - problem of key distribution & protection.

11/oct/2022
Tuesday

Transposition Technique :- Change position of letter.

1] Rainfence Technique
2] Row-Coloumn Technique

1] Rainfence Technique :-

- only change position of letter.
- we have plain text as well as key size like, 2, 3.
- Then we have to split the plain text in to the key size i.e. 2, 3 rows
- Then read 1st row then read 2nd row & ther 3rd row

For ex :-

PT = thank you very much

⟹

t		a		k		o		v		r		m		c	
	h		n		y		u		e		y		u		h

Then,

CT = takovrmchnyueyuh

- Again rainfence technigu apply on CP Text for the purpose of more Sequrity.

depth 3 CT = tkymhnyueyuhaorc as a PT.

⟹

t				h				e				a			
	k		m		n		u		y		h		o		c
		v				y				u				r	

CT = theakmnuyhocvyur

2] Row-Column Transposition :-

- size of rectangle will choose by Sender & reciver.

PT - kill Corona virus at twelve am tomorrow

Key - 43215 - choose by Sender & reciver.

Rec - 7X5

⟹

	1	2	3	4	5
1	k	i	l	l	c
2	o	r	o	n	a
3	v	i	r	u	s
4	a	t	t	w	e
5	l	v	e	a	m
6	t	o	m	o	r
7	r	o	w	x	y

CT = LNUWAOXLORTEMWIRITVOOKOVALTRCASEMRY

12/oct/2022
WED

Stream cipher means only one byte convert into cipher text, called "stream cipher"

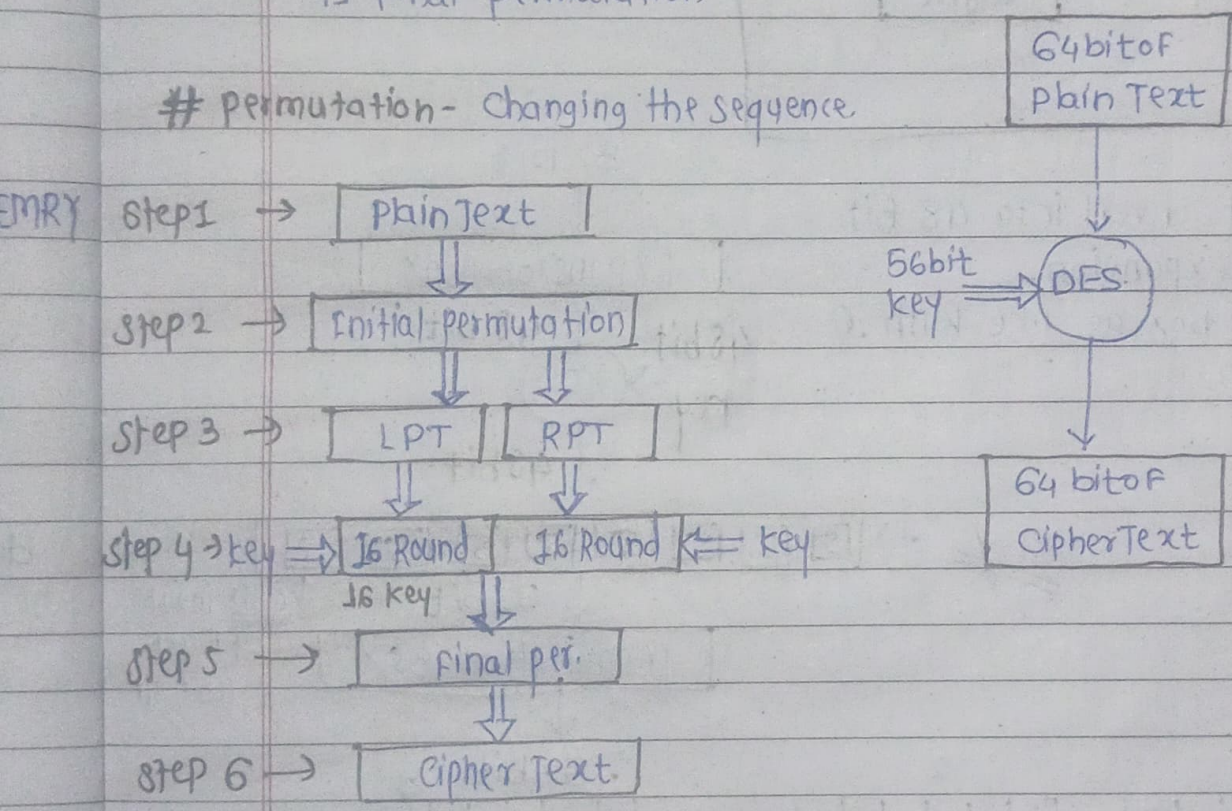
Block cipher principles :-
Whole block going to convert into cipher.

1) Number of Rouds -
2) Design of function F- How design the function.
3) key schedules Algorithm.

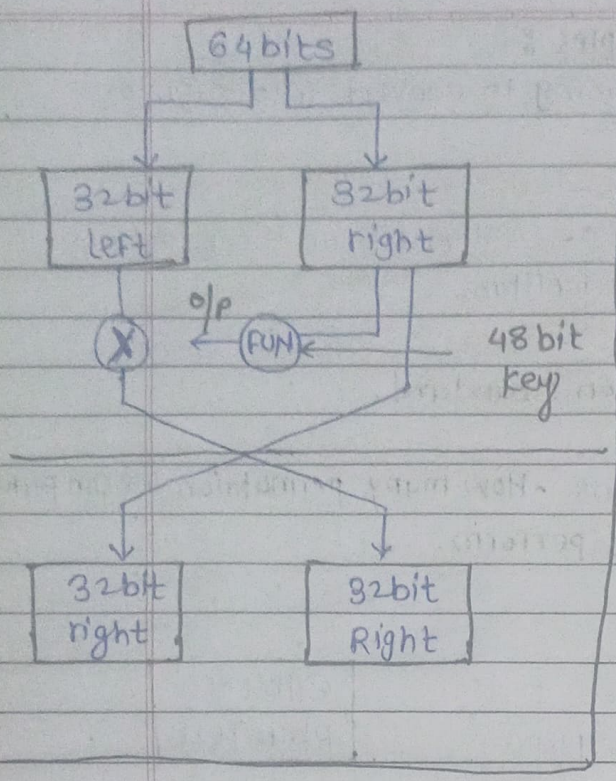
DES - Data Encryption Standard.

1] Initial permutations - How many permutaion we can perform
2] Round - 16 rounds perform.
3] Swapping
4] Final permutation

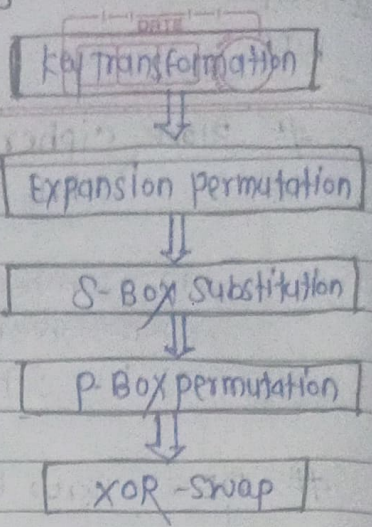
permutation - Changing the sequence.

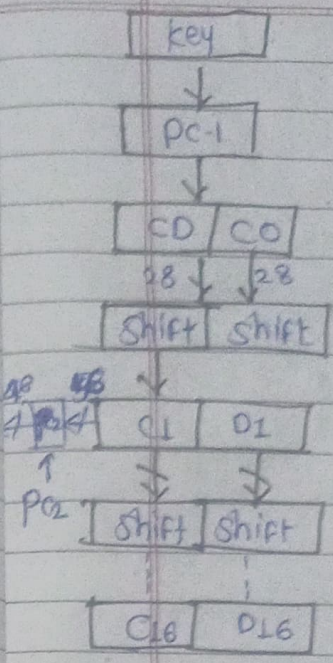


Keysize - 64 bit
56 bit
48 bit

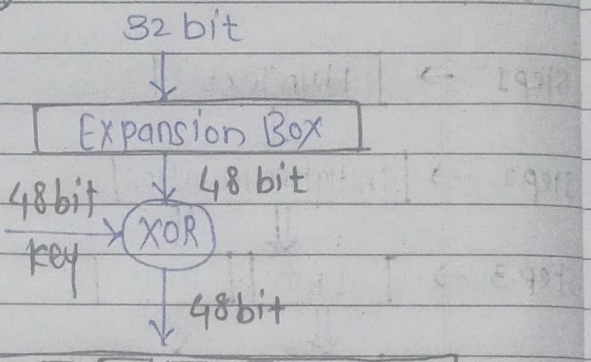


Rounds -

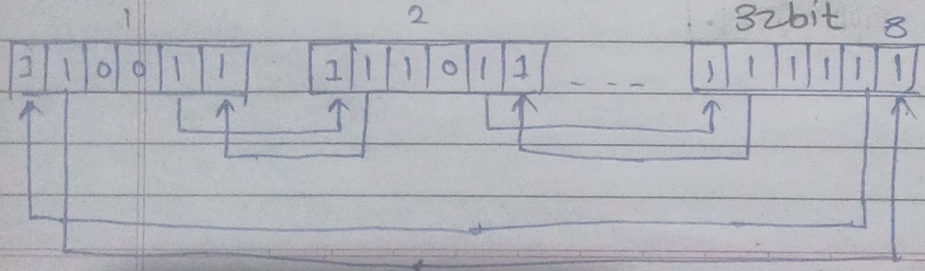




- 32 bit convert into 48 bit using expansion.
- Take 8 box as like with 6 partition.



for ex.



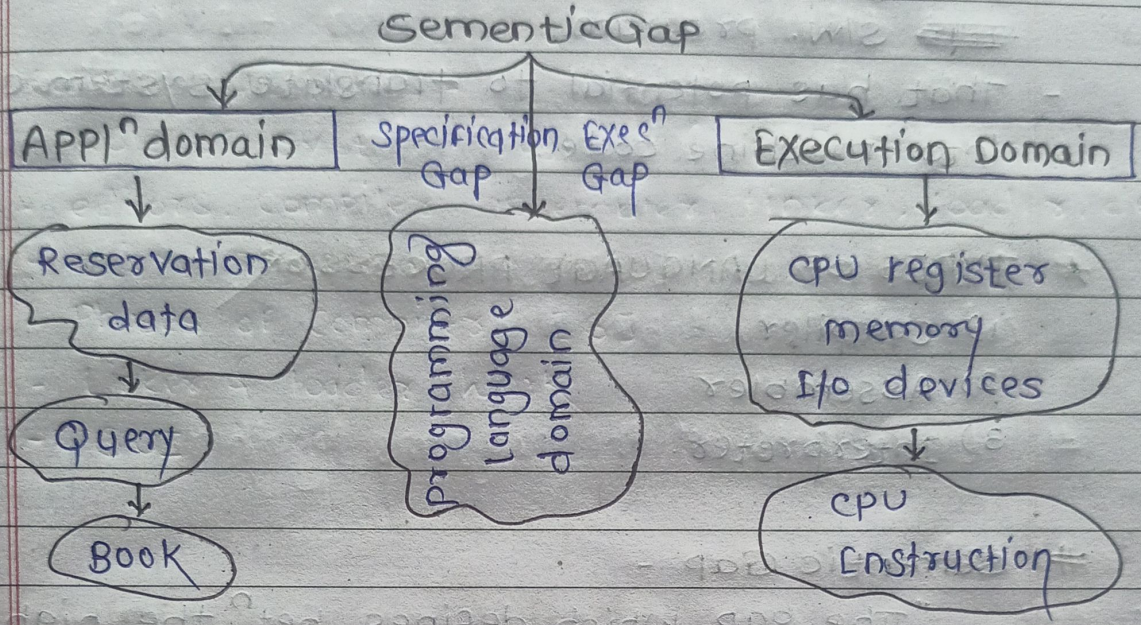
12|Sep|2022
monday

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

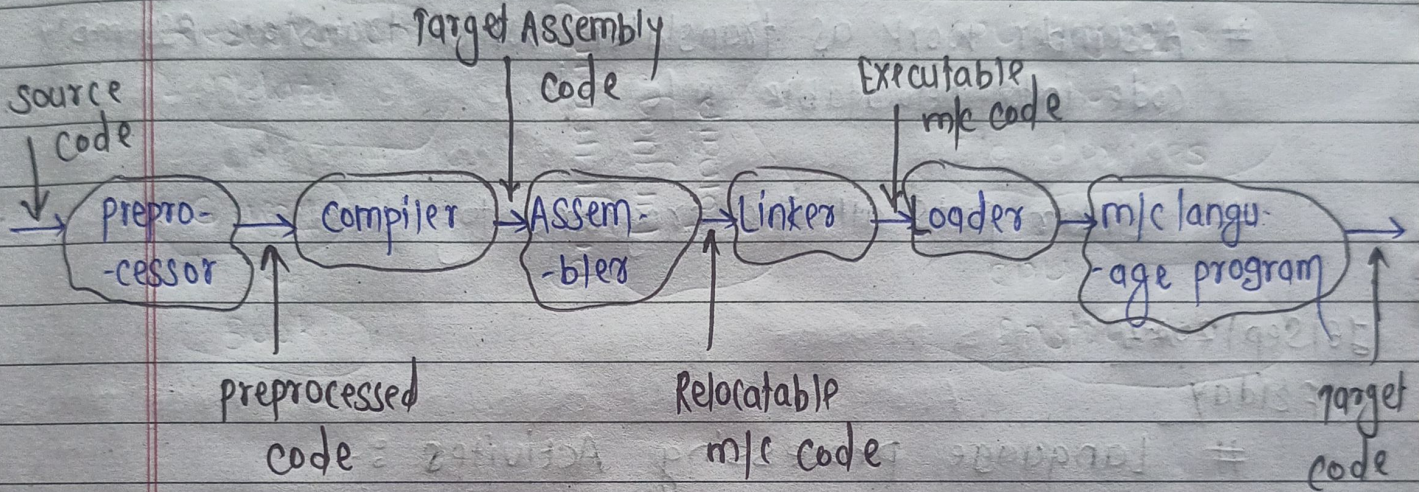


fig:- practical arrangment of language processor.

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.

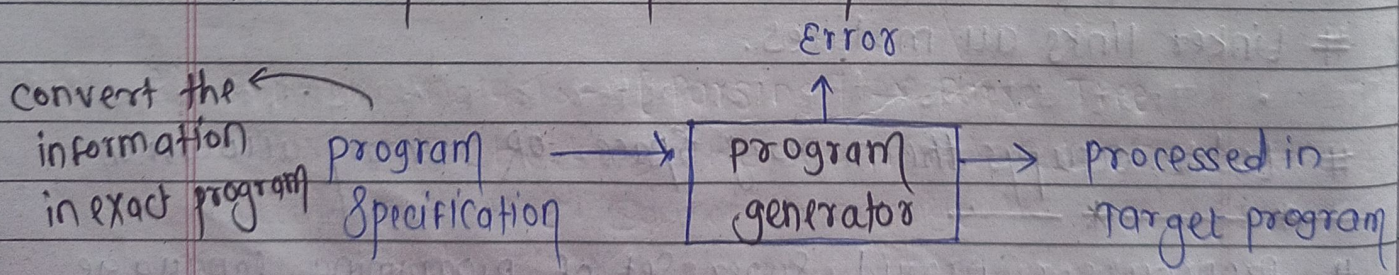
Linker puts all the programs togather.

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 :-

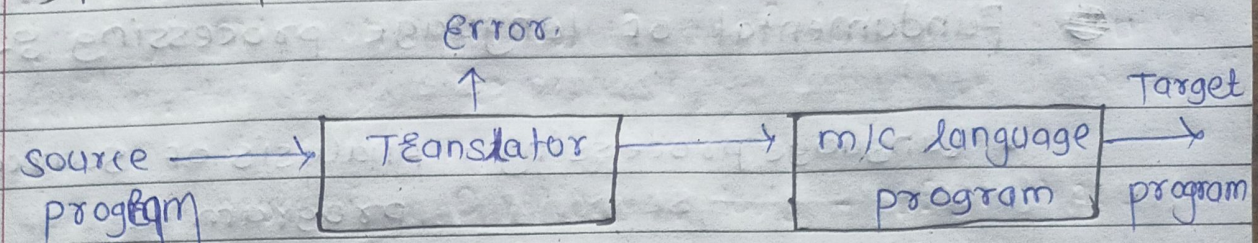


"program generator is s/w system which accept Specifications of program for generation & generates the program in Target program."

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

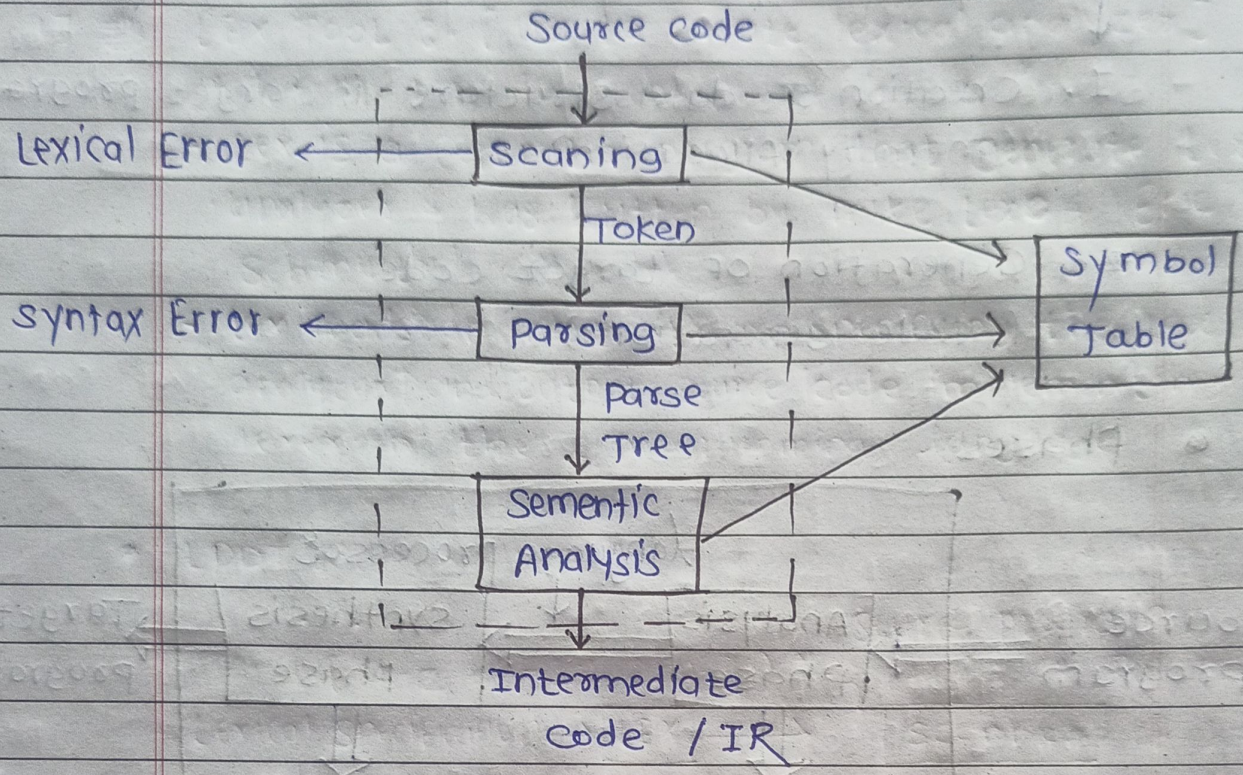
Synthesis of target program → memory allocation & Code generation is Done

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

20|sep|2022
Tuesday

IR - Intermediate Representation.

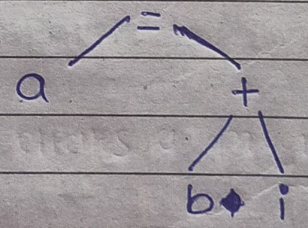
Frount End Toy Compiler :- [Analysis phase]



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]

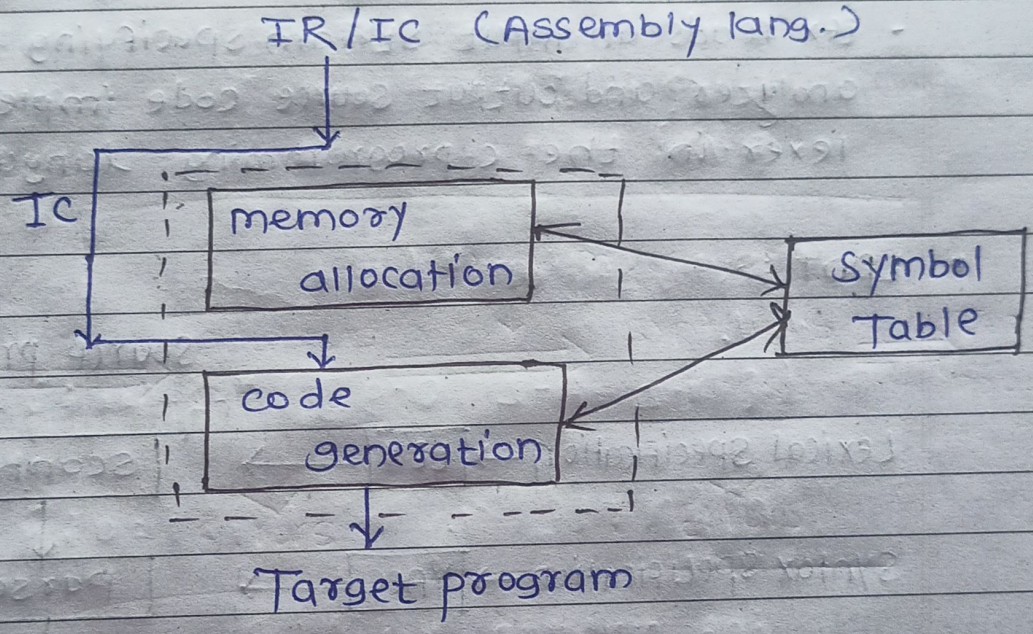


Fig. Back end toy Compiler.

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 :-

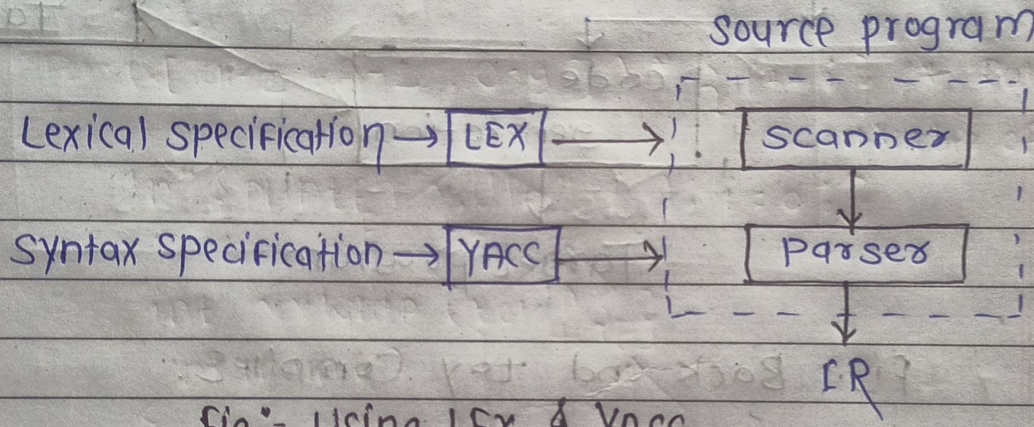
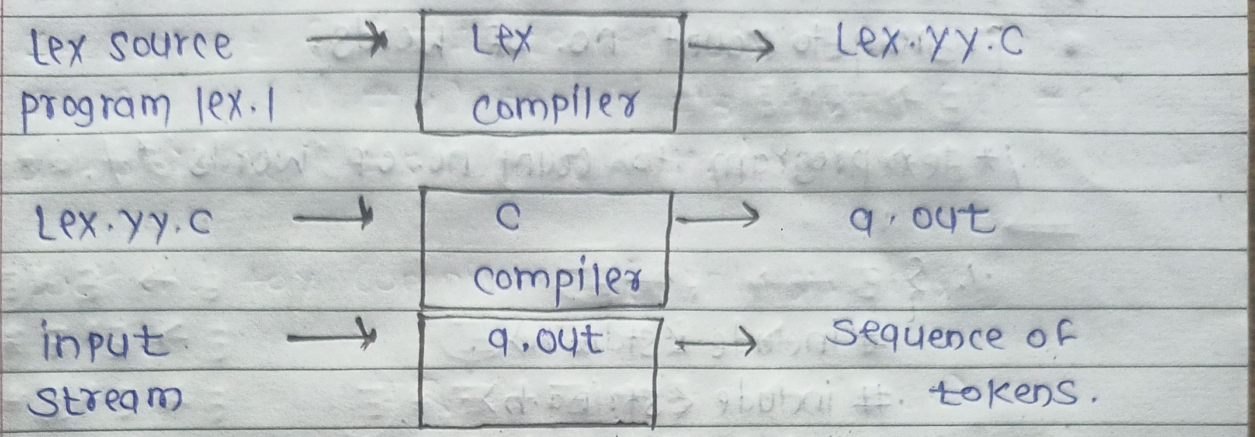


Fig:- Using LEX & YACC

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.

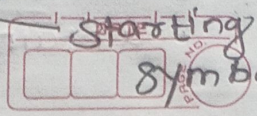
2) Translation rule :- Each of the form pattern followed by action. Each pattern is regular expression. Each action is fragment of C-code.

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,

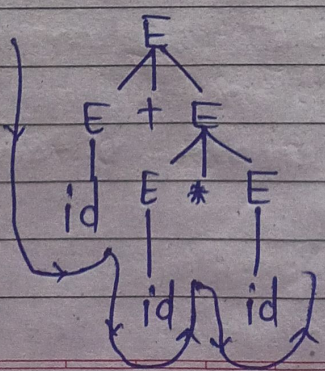
V = Veriable OR nonterminal
T = Terminal symbol
P = Production rule
S = Starting symbol

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

$E \rightarrow E+E$

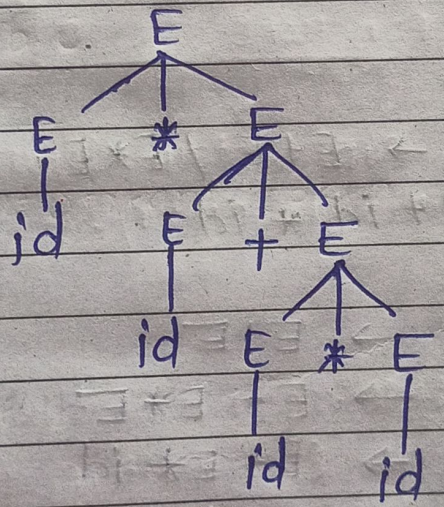
$E \rightarrow E*E$

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

$\Longrightarrow$

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

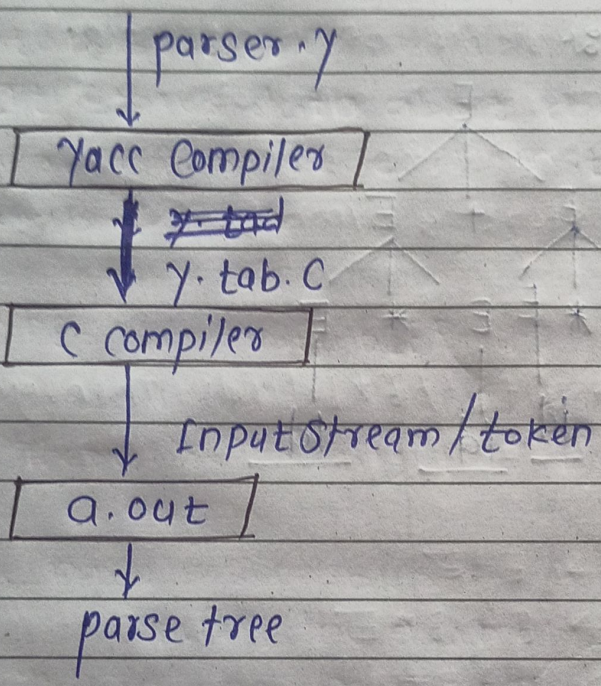
Parse Tree -



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

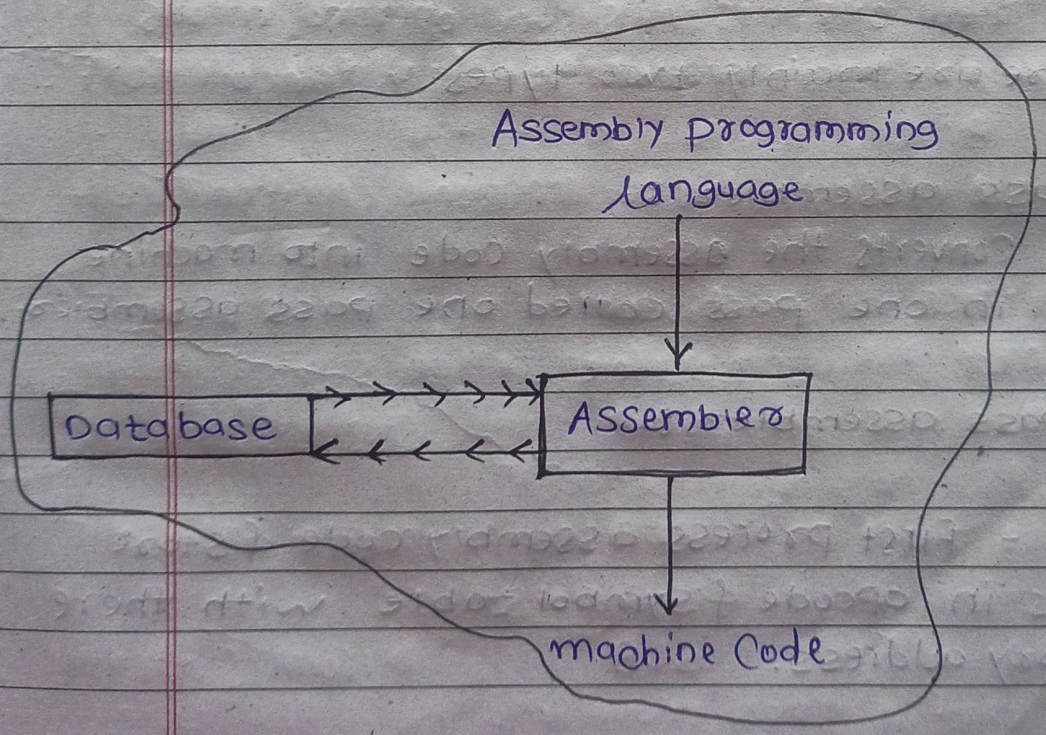
↳ Low level programming S/W OR language.

- ✓ 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>

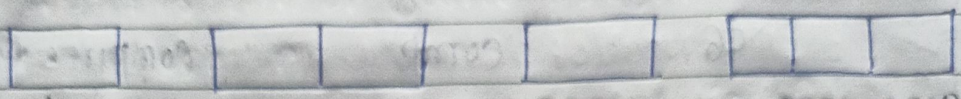
↓

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 :-

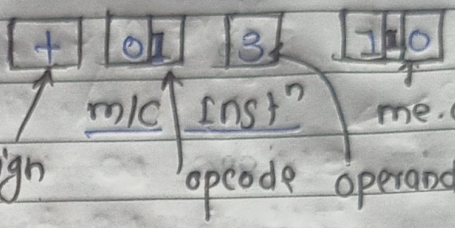
Format -



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 :-

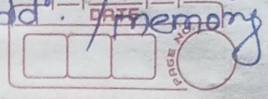
- The statement perform action during assembly / program execution called Imparative statement
- Assembly language Inst^n

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.

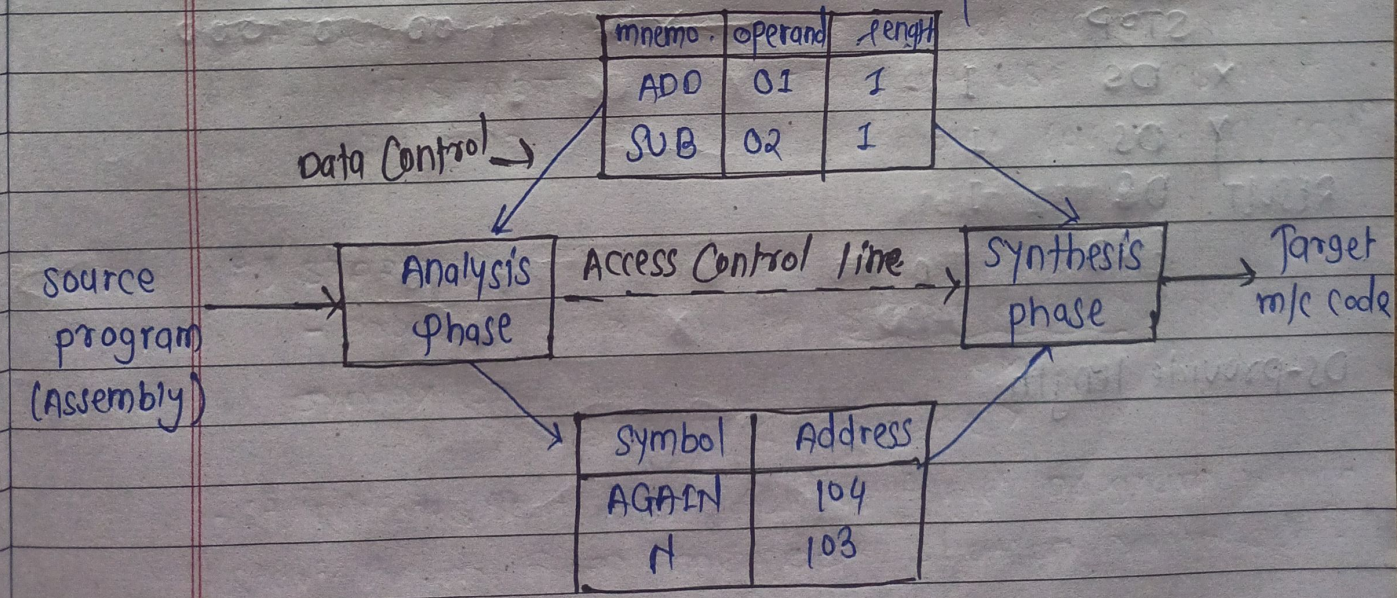


Fig :- TWO-Pass Assembler.

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 -

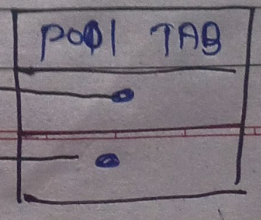
mnemonic opcode	class	mnemonic info
START	AD	R#1

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 :-

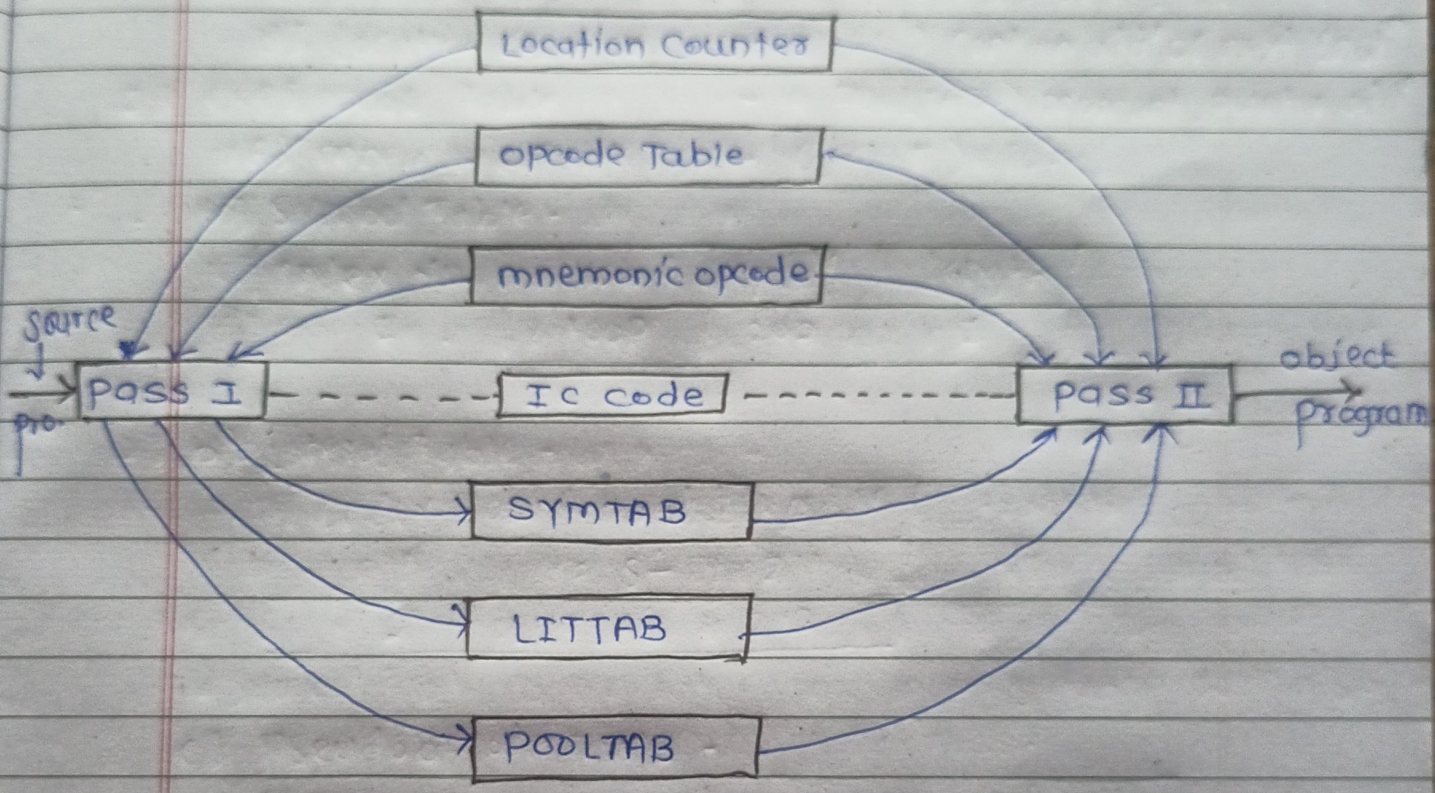
Line No.			
4	ADD	R1, X	
22	X	DC	1

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.