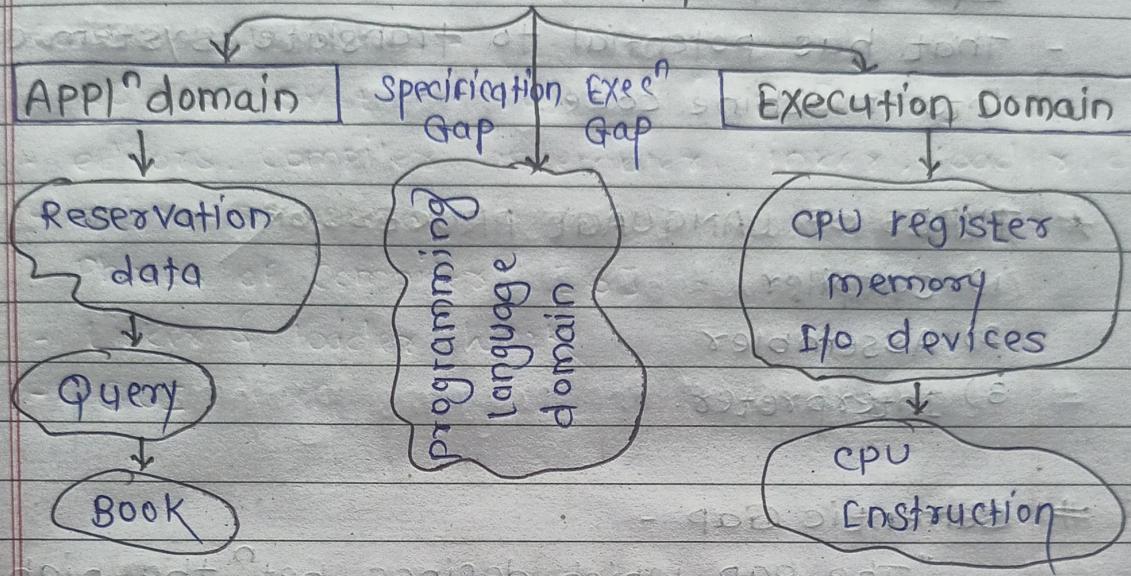


Language Processors

Language processor :-

"Bridges the gap bet" appⁿ domain & Execution domain, by using 'programming language domain'."

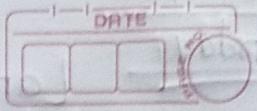
Semantic Gap

System programming :-

- system programming is the activity of programming computer system SW.
- 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 ~~SW~~ SW. program
- That has potential to translate system code into machine code.

★ TYPES OF LANGUAGE PROCESSOR

- 1) Compiler
- 2) Assembler
- 3) Interpreter.

Semantic Gap -

"The gap which defines bet' the high-level programming set and low level programming set called "Semantic Gap"."

APP1ⁿ Domain :- [Requirement analysis, planning,]

APP1ⁿ domain have isolation properties similar to operating system processes.

- multiple threads can exists Within single APP1ⁿ domain.

- An APP1ⁿ domain is a mechanism used within the common language infrastructure to isolate executed SW APP1ⁿ from one other so that they do not affect each other.

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

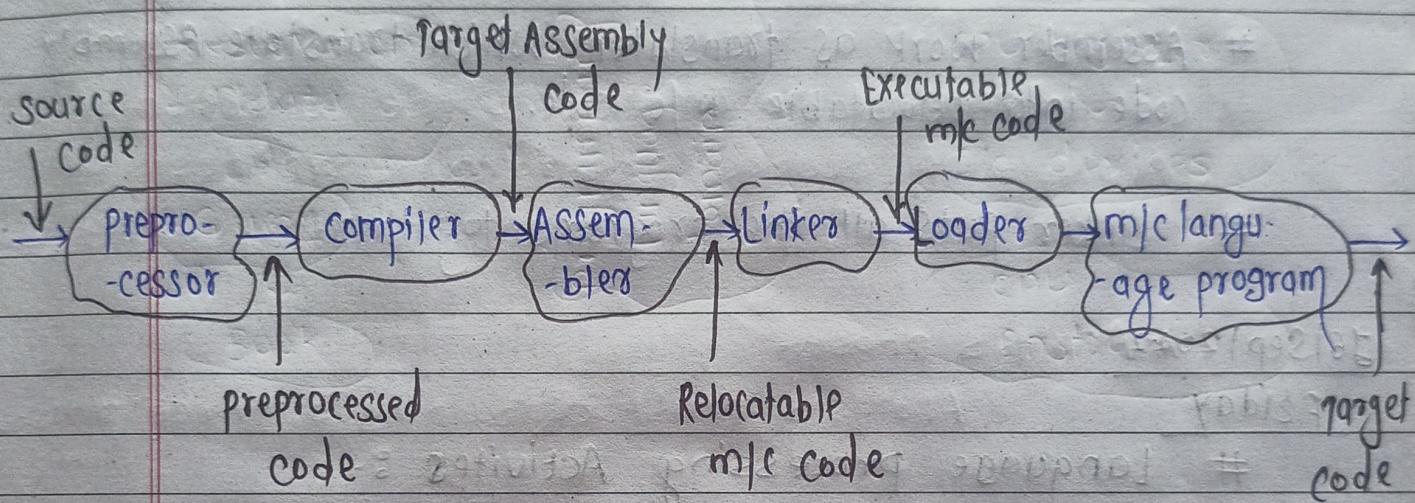


fig:- practical arrangement of language processor.

Preprocessor work for creating error free program.

Linker links all modules.

Loader occupies the memory space, OR memory allocation

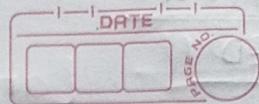
* * * * *

Preprocessor provides Super Set of programming language ~~feature~~ 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) Lexical Analyzer - symbol table generator
- 2) Syntax Analyzer - error finding
- 3) Semantic Analyzer - Parsing [grammar use] Tree
- 4) Intermediate code generator - I C Code
- 5) Code optimizer. - कमी करने की तरफ code (Reduce)
- 6) Target m/c code generator.

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

Linker puts all the programs together.

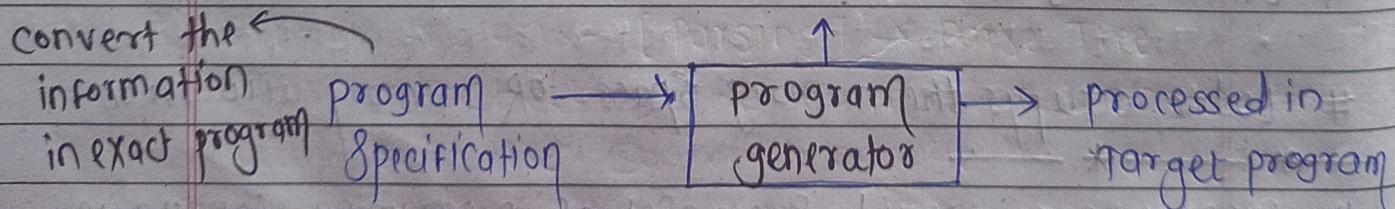
26 Sep 2022

Friday

Language processing Activities :-

- a) Program Generation [Error free program]
- b) Program Execution
- i) Program Translation
- ii) Program Interpretation

a) Program Generation Activity :-

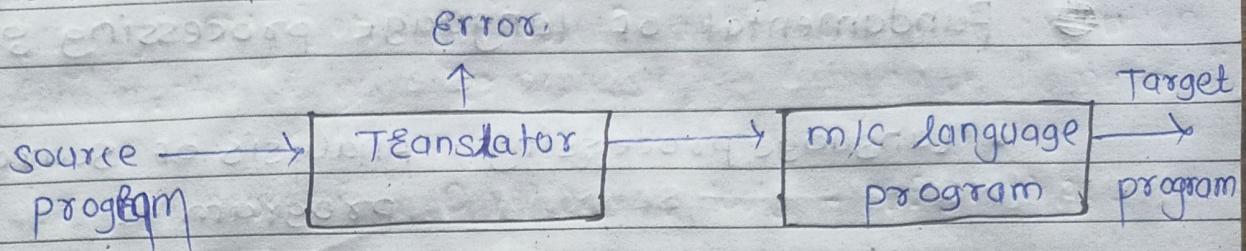


"Program generator is a software system which accept specifications of program for generation & generates the program in target program."



b) program Execution :-

i) Program Translator :-



Characteristics :-

- 1) program must be translated before execution.
- 2) Translated program may save in file, we can execute saved file OR program repeatedly.

ii) Program Interpreter :-

Three main tasks

- 1) Fetching instruction
- 2) Decoding it [convert the code in ML]
- 3) Executing it

- Interpreter reads the program & store in its memory.

Fundamental of language processing :-

1) Analysis of source program

2) Synthesis of target program

↓
memory allocation &
Code Generation is
Done

(Symbol table) creation of tokens → 1) Lexical analysis

Checking the format of Sta. → 2) Syntax analysis

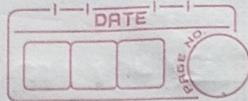
Checking the Sta. is meaningful → 3) Semantic analysis

or not

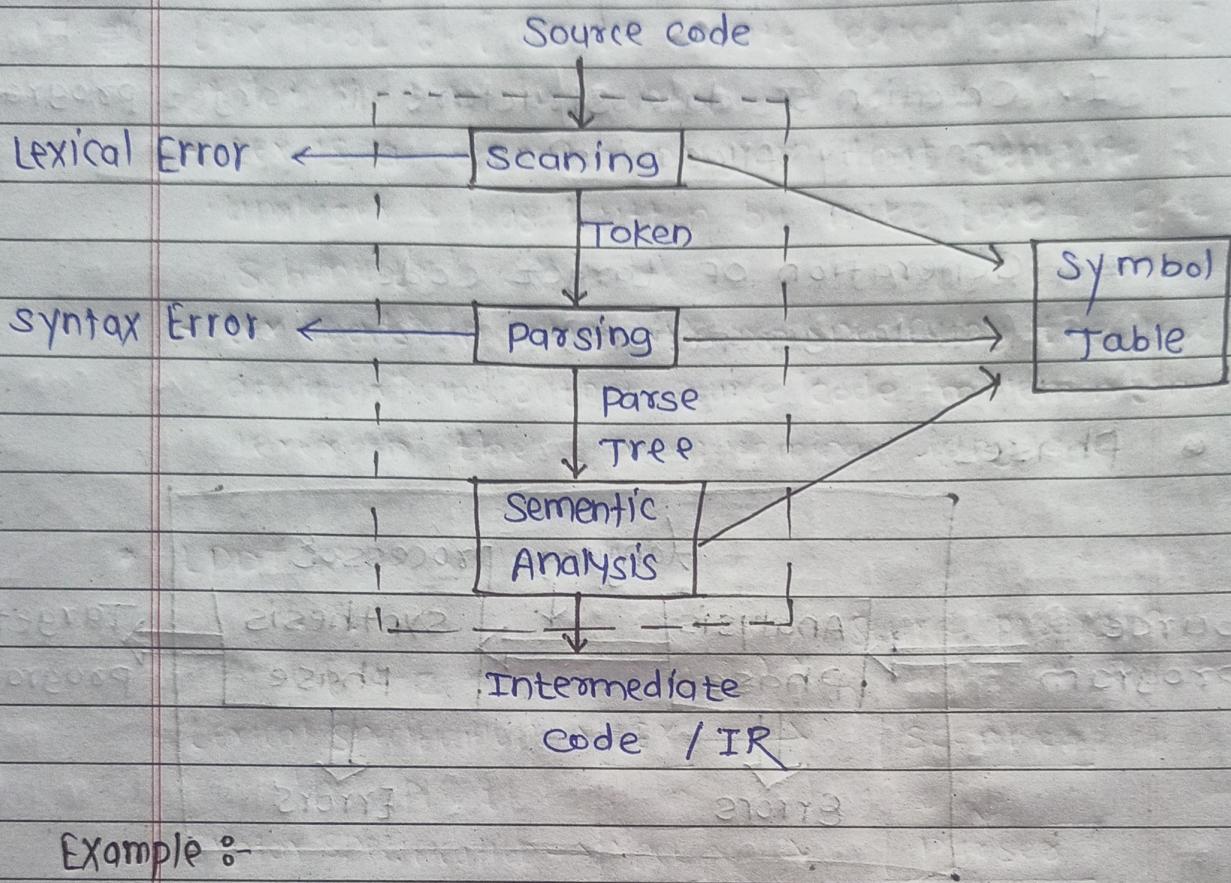
$$\text{For ex} - \text{Percent_Profit} = (\text{Profit} * 100) / \text{cost price}$$

20/sep/2022
Tuesday

IR - Intermediate Representation.



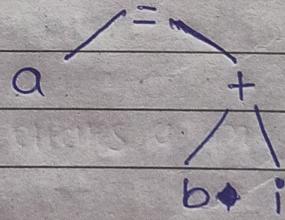
Front End Toy Compiler :- [Analysis's phase]



Example :-

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

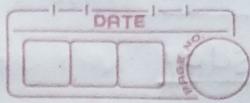
⇒



Symbol table :-

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

Loader will create .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).

Backend toy Compiler :- [Synthesis phase]

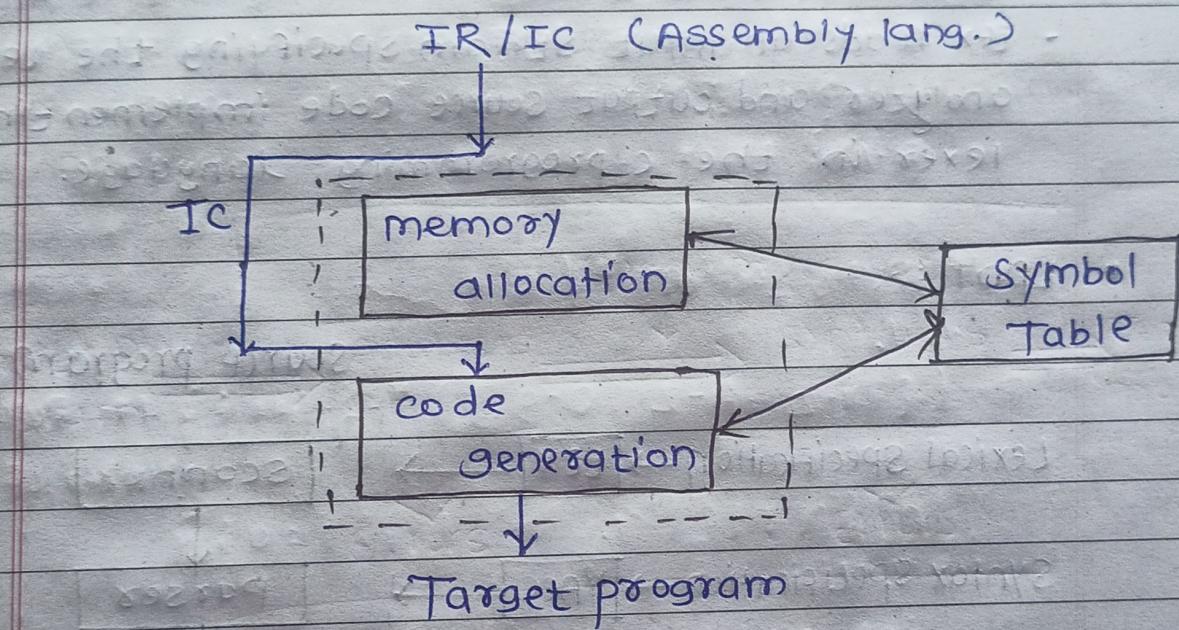


fig. Back end toy Compiler.

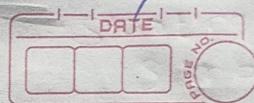
Symbol table - Maintain all information about code.

No.	symbol	Type	Length	Address
1.	a	int		2000
2.	b	real		2001
3.	c	real		2002

21 Sep 2022 (Absent)

Flex tool = LEX

Wednesday



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 Lex & Eric Schmidt.
- Lex reads an input stream specifying the lex analyzer and output source code implementing the lexer in the c programming language.

• LPDT :-

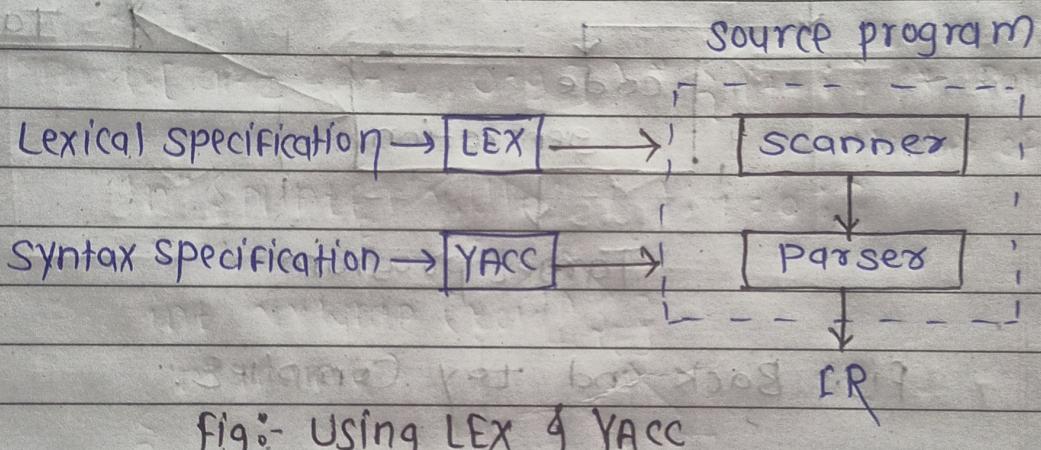
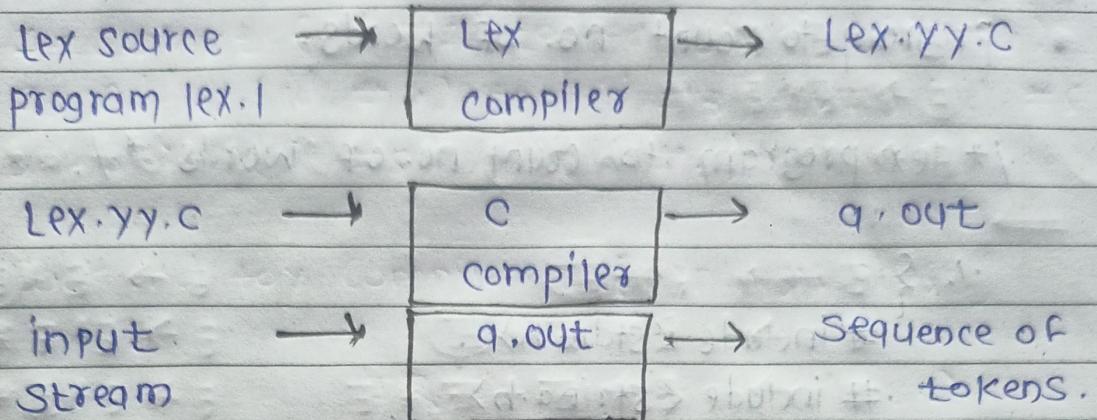
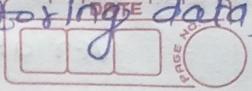


fig:- Using LEX & YACC

Function of LEX :-

- Firstly lexical analyzer creates a program lex.l in 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 maintained in symbol table [for the data]



- Content of lex program :-

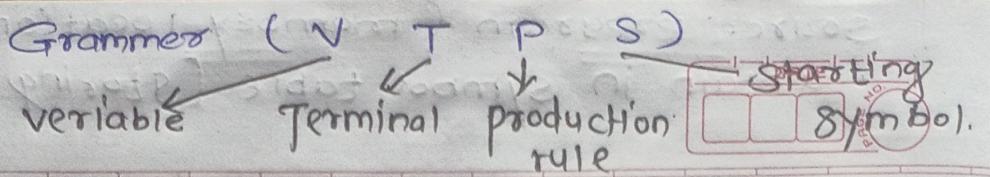
- 1) Declarations
- 2) Translation rules
- 3) Auxiliary Functions.

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 y.y. is optional. Everything in this section is copied directly to the file lex.yy.c & use in action of translation.

%. %.
divide separate



- Program to count no. of words.

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

1. {

```
#include <stdio.h>
#include <string.h>
int i=0;
1. }
```

/* Rules Section */

1. 1.

([a-zA-Z , 0-9]) * { i++ ; } /* Rule For
Number of Word */

"\n" {"printf ("y.\n", i); i=0; }

1. 1. int yywrap (void){ }

int main ()

{

// The F4

Output :-

Lex Words. |

cc lex.yy.c -lfi } Commands -

' , q , out

Vishal Dange

2

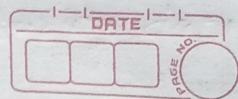
this is girl

3

28/sep/2022

WED

Grammer is a set of rule which is use for define



YACC (LPDT) Tool :- Yet another Compiler Compiler

- Developed by Stephen C. Johan.

Parse tree analysis from left to right.

Grammer :-

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

Grammer is denoted as $G = \{ V, T, P, S \}$

where,

V = Variable OR nonterminal

T = Terminal symbol

P = Production rule

S = Starting symbol

Example :-

$E \rightarrow E+E / E*E / id$ Then generate
 $id + id * id$

\Rightarrow

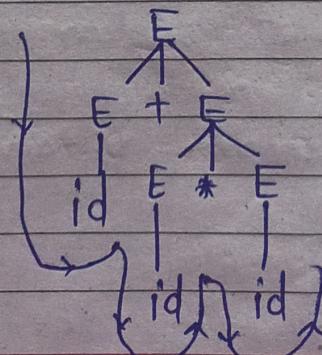
$E \rightarrow E+E$

$E \rightarrow E+E*E$

$E \rightarrow E+E*id$

$E \rightarrow E+id*id$

$E \rightarrow id*id*id$



analysis from left to right

DATE	
PAGE NO.	

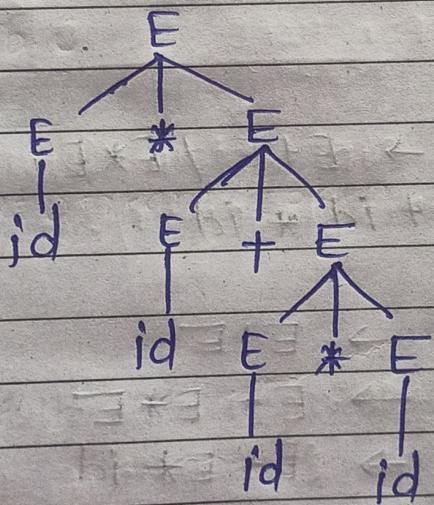
2) $E \rightarrow E+E \mid E*E \mid id$
 $E \rightarrow E+E$
 $E \rightarrow E*E$
 $E \rightarrow id$ Then generates $id * id + id * id$



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

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

Parse Tree -



Structure of YACC Compiler.

Declaration

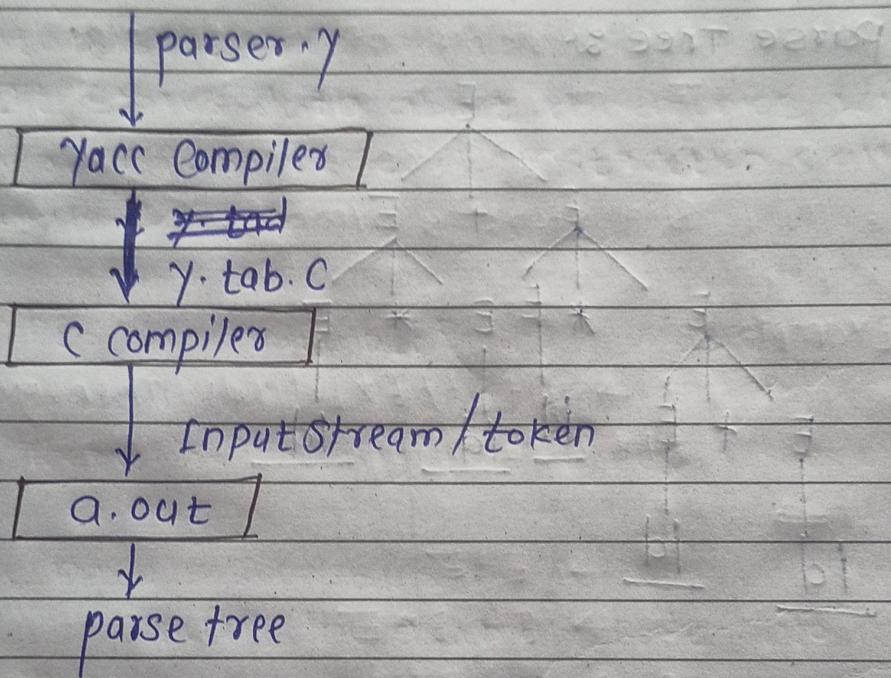
% %

Rules

% %

(Subroutine & C function.)

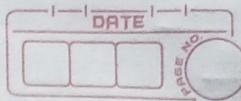
YACC Tool :-



- Firstly program will save as .y extension.
- Then yacc ~~is~~ 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.

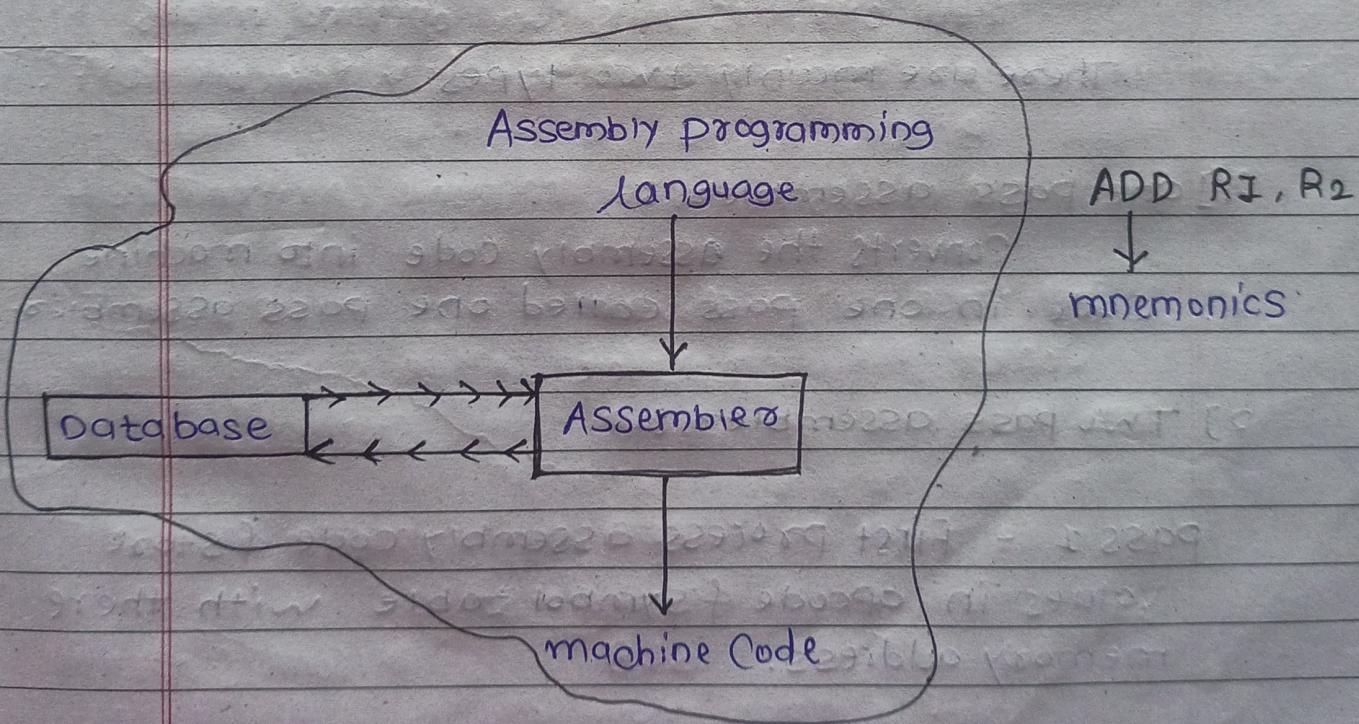
- ✓ Introduction
- ✓ Element of Assembly programming language
- A simple Assembly Scheme [mnemonic]
- Pass structure of Assemblers
 - a) one pass Assembler - Whole code compile.
 - b) TWO pass Assembler
 - [opcode table & symbol]
 - Target m/c



Introduction :-

"Assembler is a SW that converts assembly language code into machine code."

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



- * Assembly language is low level programming language, & It gives instⁿ to processor for performing different tasks.

Symbolic representation of Instruction

or opcode

Opcode of mnemonics :-

↓
Binary code representation of instruction

Specific information of instruction which takes.

For ex.

ADD A,B

Where,

ADD - is the mnemonics
& A,B are operands

• Database :-

"It Stores data about mnemonics with their binary code of instruction called database."

TYPES OF ASSEMBLER :-

These 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 their memory address.

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

07/10/2022

Monday

Use to Create Assembly

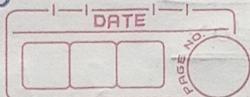
DATE

Element of Assembly programming language :- program

- 1) Format of Assembly Program Statement
- 2) Simple Set of Instructions
- 3) m/c Instruction Format
- 4) Simple assembly program
- 5) Type of Assembly language statement.

Format of Assembly program Stat.

- a) mnemonic operation code - symbolic representation.
 - b) Symbolic operand - represent data OR Instⁿ
 - c) Data Declaration - number system
- operation code for machine instⁿ & use numerical operation code if ~~not~~ easy to remember.
- programmer can associate symbolic name with data & instⁿ & use symbolic name as operand in assembly stat.
- Data can declared varietys of notation ~~not~~ including decimal notation.



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. if second is memory word with symbolic name.

Example :-

ADD AREG ONE



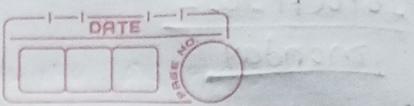
Where ADD is the mnemonic

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	Assembly mnemonic	Remark
00	STOP	Stop execution
01	ADD	perform addition
02	SUB	perform subtraction
03	MUL	perform multiplication
04	MOVER	move from memory to registers



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 registers

10 PRINT print Content of reg.

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

Format \Rightarrow BC <condition code specification>, <memory Address>

LT - Lower than

LE - Lower than Equal to

EQ - Equal TO

GT - Greater Than

GE - Greater than equal to

ANY - Unconditional Control transfer

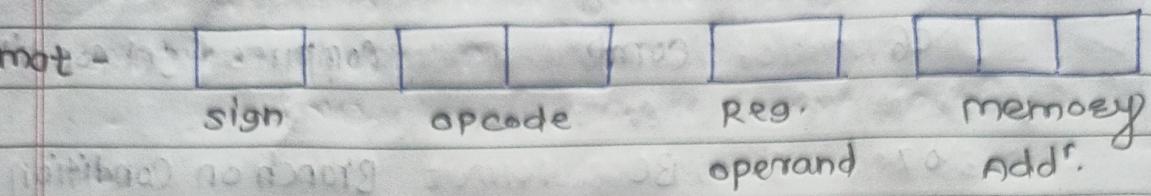
10/10/2022

Monday



3] Machine Instruction Format :-

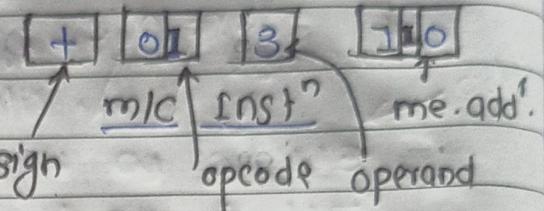
the Format -



For ex.

ADD CREG, ONE

Assembly Instⁿ



- The opcode, reg. operand, memory add', occupies two, one or three digits respectively.

4] TYPES of Assembly language statement :-

There are three types -

- A) Imperative Statement - Assembly Instⁿ
- B) Declarative Statement
- C) Assembly Directives.

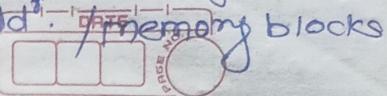
A] Imperative Statement :-

- The statement perform action during assembly / program execution called Imperative Statement.
- Assembly language Instⁿ

Ex.

MOVE R CREG, BREG.

Content of BREG move into the CREG. using the MOVR Instⁿ called imperative instⁿ. / statement

Constant value always
represent add^r. 

B) Declarative Statement :-

i) Declarative storage

ii) Declarative constant

i) Declarative storage :-

Syntax -

[Label] DS <constant>

It reserves area of memory & associate symbolic name.

Ex. ABC DS 200

ii) Declarative Constant -

Syntax -

[Label] DC '<value>'

It construct memory word or memory add^r.
Containing constant

Ex. XYZ DC 'I'

c) Assembly Directives -- It cannot generate m/c.

i) START

- They are use to instruct

ii) END.

Assemblers for action / inst^r.

i) START :-

- Program start with START <200> inst^r.

Syntax :-

START <constant>

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

ii] END -

- It indicates END of Source program. (3)

Syntax -

END.

5] Simple Assembly program :-

		location ↓ Counter	sign	opcode	reg operand	addr
START	I01					
READ	N	I01>	+	09	0	113
MOVER	BREG, ONE	I02>	+	04	2	115
MOVEM	BREG, TERM	I03>	+	05	2	116
MUL	BREG, TERM	I04>	+	03	2	116
MOVER	CREG, TERM	I05>	+	04	3	116
ADD	CREG, ONE	I06>	+	01	3	115
MOVE M	CREG, TERM	I07>	+	05	3	116
COMP	CREG, N	I08>	+	06	3	113
BC	LE , AGAIN	I09>	+	07	2	104
MOVEM	BREG, RESULT	I10>	+	05	2	114
PRINT	RESULT	I11>	+	10	0	114
STOP		I12>	+	00	0	000
DS	1	I13>				
DS	1	I14>				
DC	'1'	I15>	+	00	0	001
DS	1	I16>				
END						

22 Oct 2022
WED

5 Data Structure, present in
assembler.



A simple assembly Scheme :-

- 1] Design Specification for an Assemblies.
- 2] Phases of assemblies
- 3] Data structure used by assembler. [Storing & Fetching Data]

1] Design Specification for an Assemblies :-

- 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.

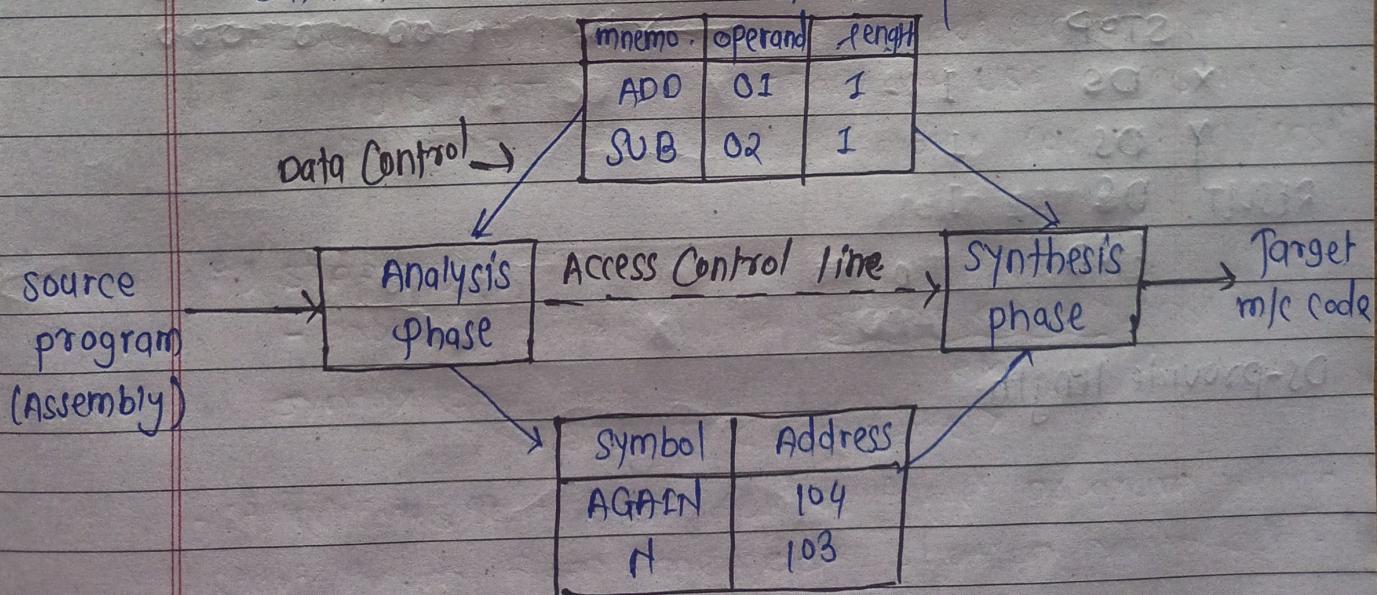
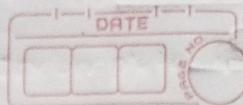


Fig: Two-Pass Assembler.



a) Analysis phase :-

- symbol table building
- memory allocation
- Location Counter [info. about next Instⁿ] & Add^r

b) Synthesis phase :-

- obtain opcode from mnemonic table.
- obtain memory add^r from symbol table
- synthesis machine Instⁿ.

Addition program :-

START	101	- Lc	m/c Inst ⁿ
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 I

Y DS I

RESULT DS I

END

DS-provide length.

3) Data structures used by Assemblers :-

- 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	Addr'	length
1	loop	202	1

b) Literal Table -

Index No	literal	Addr'
1	= 'S'	211
2	= 'I'	212

c) Pseudo opcode Table -

mnemonic opcode	class	mnemonic info
START	AD	R#I

d) Location Counter -

variable	Addr'
X	108
Y	109

e) Pool Table -

literal	Addr'
= 'S'	100
= 'I'	101

pool TAB

Control pool

Transfer

17/10/2022

monday

DC - Declarative constant

DATE

TII - Table of Incomplete inst?

Pass structure of Assembler :-

1] Single pass Assembler

2] Two Pass Assembler.

Single pass Assembler -

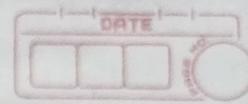
- Assembler ~~will~~ scan input file in once
- Generate target machine code
- It is a faster than multipass Assembler
- It contain symbol, literal, mnemonic & opcod table as data structure
- In that forward reference problem arises it means a variable is found before define To solve this forward reference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

Example :-

LineNo.	OpCode	Value	OpCode	Value
4	ADD	R1, X		
1				
1				
1				
22	X	DC	1	

* TII Table

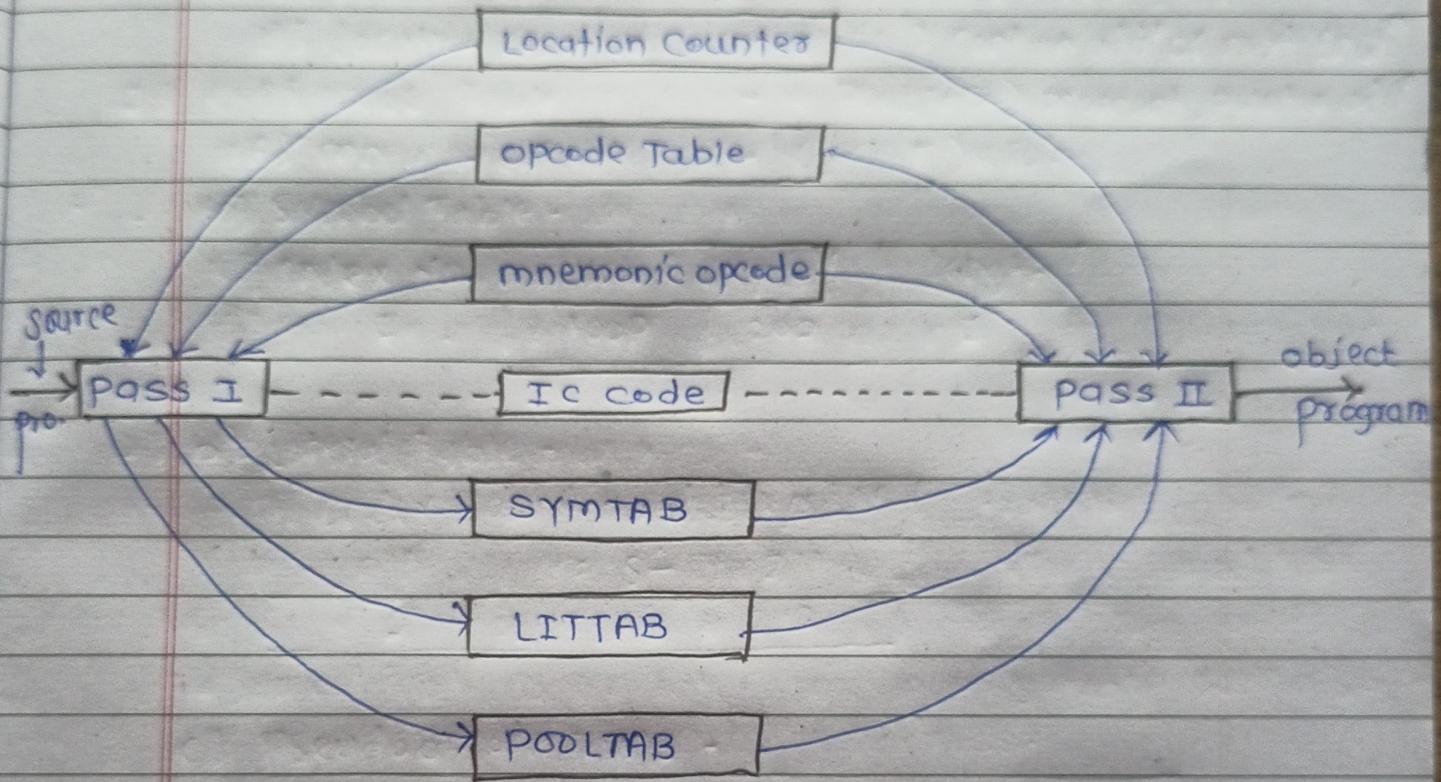
Type	Value	Line No
ADD	X	4



TWO pass / multipass Assemblies :-

I] pass I = [Analysis]

II] pass II = [synthesis phase]



I] pass - I :-

- Separate different data structures. OPTABLE, MOT etc.
- Built SYMTAB & LITTAB.
- Perform Location Counter processing
- Construct intermediate code.

II] pass - II :-

- Synthesis of target program.
- Execute file & generate a file.

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?