23/08/2023

220 / 145

8 Tuesday

* Compiler Design *

2017 AUGUST

Week-32

SCHEDULES

8.00

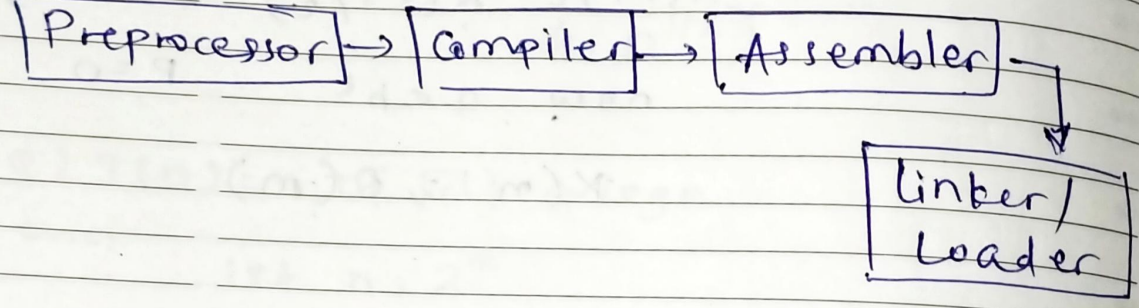
Translator - It is a software that converts high level language to low level of language

9.00

10.00

Internal Architecture of Translator

11.00



12.00

1.00

2.00

3.00

Interpretor	compiler
4.00 i) Compile one line at a time	i) compile whole program at a time
5.00	
6.00 ii) Takes more time.	ii) faster than interpretor.
7.00	
iii) eg python, javascript	iii) eg. C, C++

8.00

AUG - 2017

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

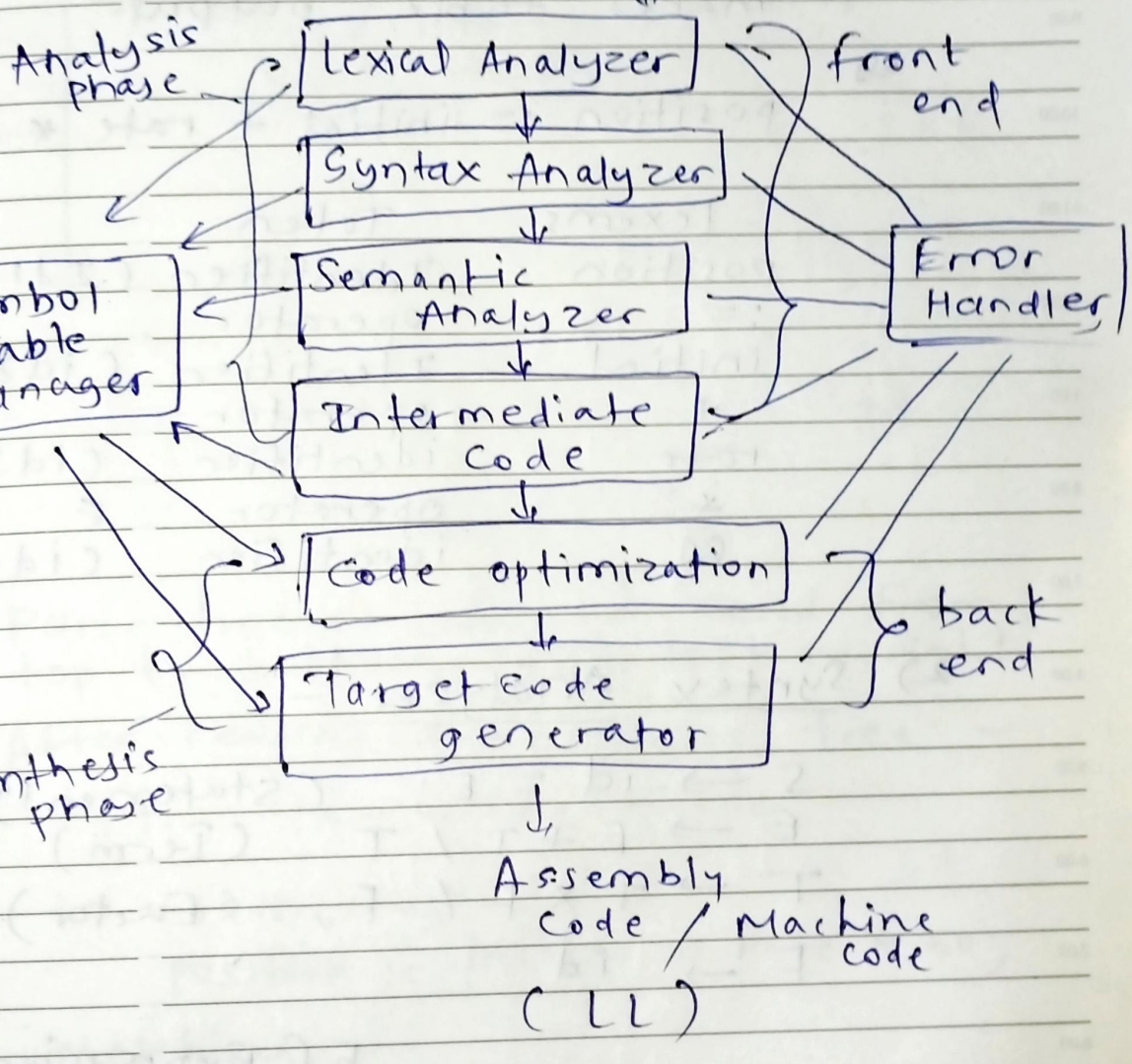
SEP - 2017

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

NOTES

SCHEDULES

Phase of Compiler :- HLL



9.00
10.00
11.00
12.00
1.00
2.00
3.00
4.00
5.00
6.00
7.00
8.00

NOTES

AUG - 2017

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

SEP - 2017

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	

SCHEDULES

8.00

9.00

10.00

11.00

12.00

1.00

2.00

3.00

4.00

5.00

6.00

7.00

8.00

1) Lexical Analyser -

it finds out lexems (variables, keywords), tokens

it makes table output.

eg.

position := initial + rate * 60;

Lexems	Token
position	Identifier (Id1)
:=	operator
initial	Identifier (id2)
+	operator
rate	identifier (id3)
*	operator
60	identifier (id4)

2) Syntax Analyzer -

S → id = E; (Statement)

E → E + T / T (Term)

T → T * F / F (Factor)

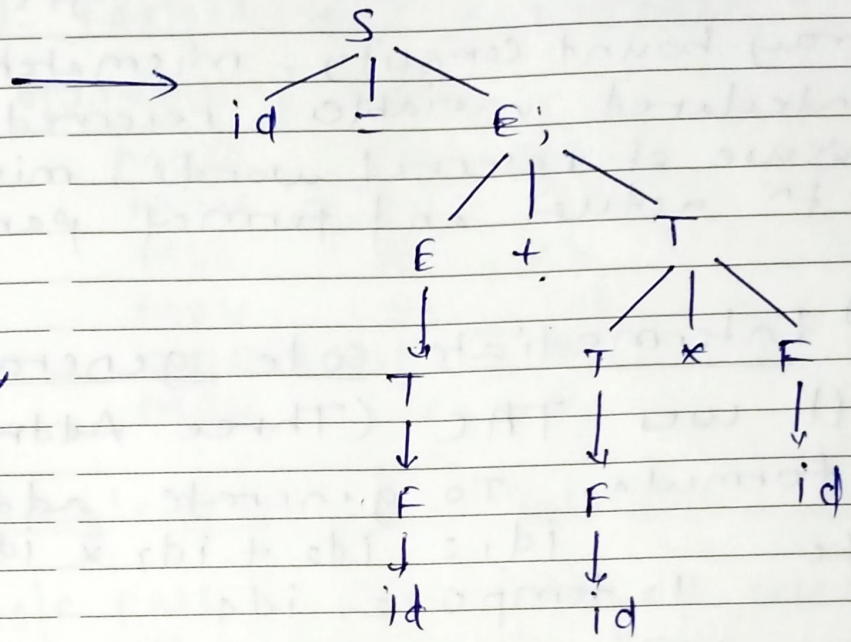
F → id

E (Expression)

SCHEDULES

8.00
9.00
10.00
11.00
12.00
1.00
2.00
3.00
4.00
5.00
6.00
7.00
8.00

Parse tree



Parse tree is always read from top to bottom and left to right.

After reading above parse tree -

$id_1 = id_2 + id_3 * id_4$

ie same as

position := initial + rate * 60;

Matching

NOTES

AUG - 2017

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

SEP - 2017

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	
24	25	26	27	28		

SCHEDULES

3) Semantic Analyzer -

It is responsible for type checking, array bound capacity, mismatch error, undeclared variable, reserved ~~or~~words, misuse of reserved words, mismatch bet^n actual and formal parameters.

4) Intermediate code generator -

It use TAC (Three Address Code) formula. To generate addresses. like

$id_1 = id_2 + id_3 * id_4$

1. temp0 = id4
2. temp1 = id3 * temp0
3) temp2 = id2 + temp1
4. temp3 = temp2

5) Code optimization -

temp1 = id2 * 60

temp2 = id1 + temp1

(id3)

AUG - 2017

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

SEP - 2017

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

NOTES

SCHEDULES

6) Target code generator -

Assembly code

```
MOV   R1 , 60
MOV   R2 , id2
MUL   R1 , R2
MOV   R3 , id1
ADD   R3 , R1
MOV   id3 , R3
```

* Single pass & Multi pass compiler -

- Single pass - compiler will pass through all phases only once
- Multipass - compiler will pass through all phases multiple time and producing intermediate code every time.

Multipass compiler generates machine dependent as well as machine independent code.

NOTES

SCHEDULES

8.00 ★ Cross compiler - when the compilation of source code is done on one system and then it is run on other system then it is called cross compiler.

9.00

10.00

11.00

* for the lexical analyser use the 'lex' tool.
* for semantic analysis use directed translat^ enginec
* for intemediate code generat^ use automatic code generator
* for code optimizer use data flow analysis engine
* for target code generat^ use compiler construct^ tool kit.
* for syntax analyzer use yet another compiler

12.00

1.00

2.00

3.00

4.00

5.00

6.00

7.00

8.00

AUG - 2017

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

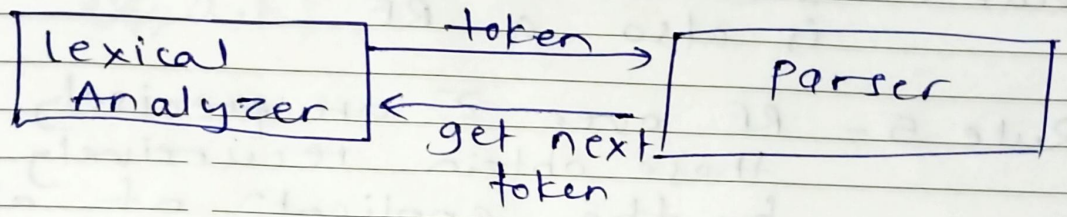
SEP - 2017

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

NOTES

SCHEDULES

2nd chapter - Role of lexical Analyzer -



Also it generates error message.

* Regular Expression - are used for representing certain sets of strings in an algebric form. ~~So wh~~

Rule 1 - Any terminal symbol i.e ϵ, Σ including empty (Λ) and null (ϕ) are regular expressions.
eg. a, b, c, a1, ϵ, Σ, dz

Rule 2 - union of two R.E is also a regular expression

Rule 3 - concatenat^ of two RE is also a RE.

NOTES

AUG - 2017

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

SEP - 2017

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

SCHEDULES

Rule 4 - the iterat^n or closure of a RE is also a RE

Rule 5 - RE over Σ are precisely those obtain recursively by the applicat^n of above rules once or several times.

1) $\{0, 1, 2\}$ - $0+1+2$

2) $\{\wedge, ab\}$ - $\wedge ab$

3) $\{\wedge, 0, 00, \ldots\}$ - 0^*

4) $\{0, 00, 000, \ldots\}$ - 0^+

Q) $S \rightarrow AaAb \mid BbBa$

$A \rightarrow \varepsilon$

$B \rightarrow \varepsilon$

AUG - 2017

S	M	T	W	T	F	S
		1	2	3	4	5
6	7	8	9	10	11	12
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31		

SEP - 2017

S	M	T	W	T	F	S
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

NOTES