Computer Science and Engineering Discipline

Khulna University

Compiler Design Lab 01

LEXICAL ANALYSIS

- The first phase of compilation.
- Takes raw input, which is a stream of characters, and converts it into a stream of tokens, which are logical units, each representing one or more characters that *belong together*.

Typically,

1. Each keyword is a token.
2. Each identifier is a token
3. Each constant is a token
4. Each sign is a token

Scanning

The raw input is processed in certain character-by-character ways, such as

- Remove comments.
- Replace strings of tabs, blanks, and newlines by single blanks.

The Token Output Stream

Before passing the tokens further on in the compiler chain, it is useful to represent tokens as pairs, consisting of a

1. Token class (just *token* when there is no ambiguity), and a
2. Token value.

The reason is that the parser will normally use only the token class, while later phases of the compiler, such as the code generator, will need more information, which is found in the token value.

INPUT

A raw source program.

Example

```
int main ()
{
        int a, b, sum;
        sum =  a+b;
        return 0;
}
```

OUTPUT

- **Tokens**

Lexemes	Token Name	Attribute Value
int	int	-
main	identifier	-
(	special symbol	opening braces
)	special symbol	closing braces
{	special symbol	left curly braces
int	int	-
a	id	pointer to symbol table entry
,	special symbol	comma
b	id	pointer to symbol table entry
,	special symbol	comma
sum	id	pointer to symbol table entry
;	special symbol	semicolon
sum	id	pointer to symbol table entry
=	operator	assignment
a	id	pointer to symbol table entry
+	opearator	addition
b	id	pointer to symbol table entry
;	special symbol	semicolon
return	return	-
0	number	constant
;	special symbol	semicolon
}	special symbol	right curly braces

- **Symbol Table**

Symbol	Token	Data Type	Pointer to Symbol Table Entry
main	id	-	0
a	id	int	1
b	id	int	2
sum	id	int	3