Date / /

* Lexeme is a meaningful sequence of characters in the source program.

* Pattern is a rule of description.
eg: A regular expression of identifier - $l(l+d)^*$
A regular expression of delimiter - $d(d)^*$

Input Buffer

* Let us examine some ways that the simple but important task of reading the source program can be speeded.

* This task is made difficult by the fact that we often have to look one or more character beyond the next lexeme before we can be sure we have the right lexeme.

* We, thus, introduce a 2 buffer scheme that handle large look ahead safely.

* We then consider an improvement involving sentinal that save time checking for the end of input buffer.

Buffered Pair Algorithm

* The amount of time taken to process character and large no of characters that must be processed during the compilation of large source program. Specialised buffering technique have been developed to reduce the amount of overhead required to process a single input

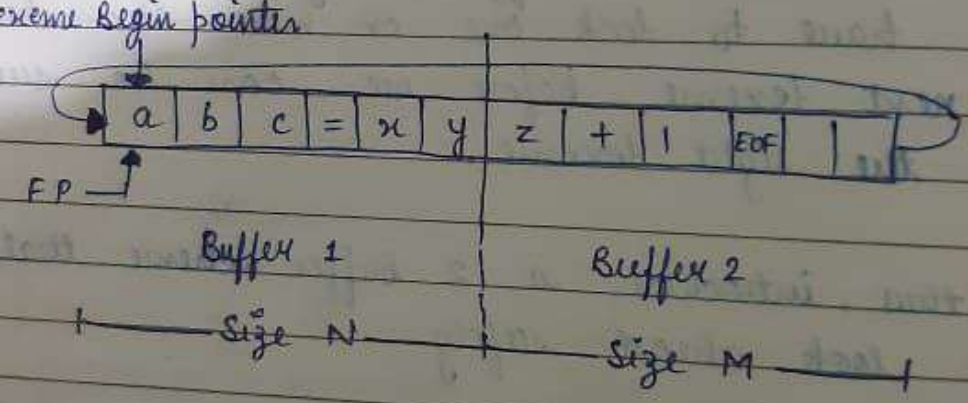
character.

* An important scheme involves 2 buffers that are reloaded again & again.

* Two pointers are maintained -

Lexeme Begin pointer : It marks the beginning of the current lexeme which we are attempting to determine

Forward Pointer : It scan ahead until a pattern match is found.



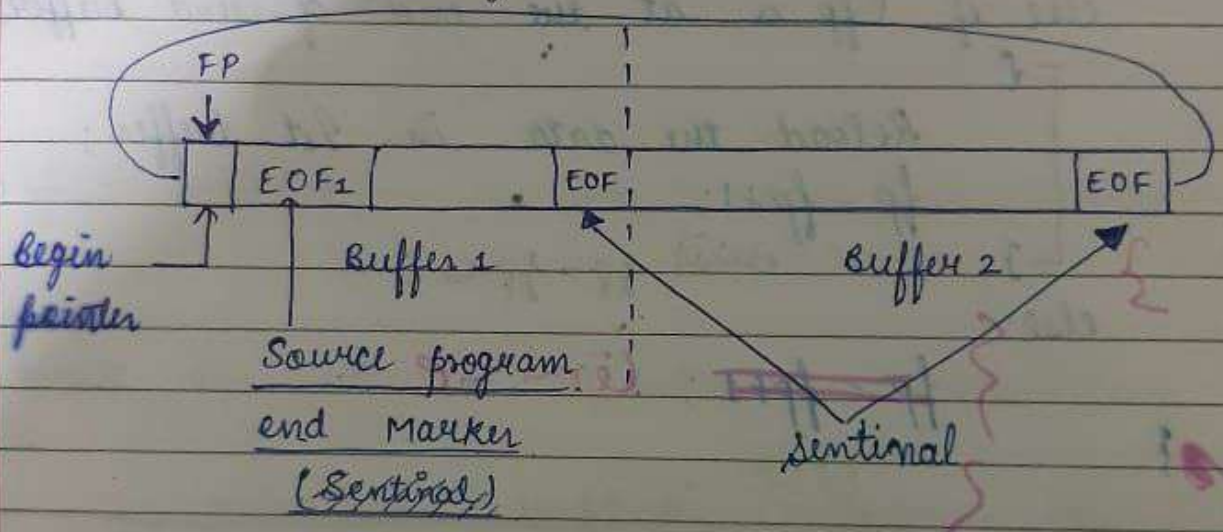
Code to advance the forward pointer

```
if (fp is at the end of 1st half)
{
    Reload the second half
    fp = fp+1;
}
else if ( fp is at the end of 2nd half)
{
    Reload the 1st half (Begin 1st Buffer)
    fp = fp+1;
}
```

```
else if (fp == EOF)
        return ;
else
        fp = fp+1;
```

5/Feb/18

Buffer Pair Algorithm with sentinal element



We make two test - one for the end of the buffer
- one to identify what character is read.

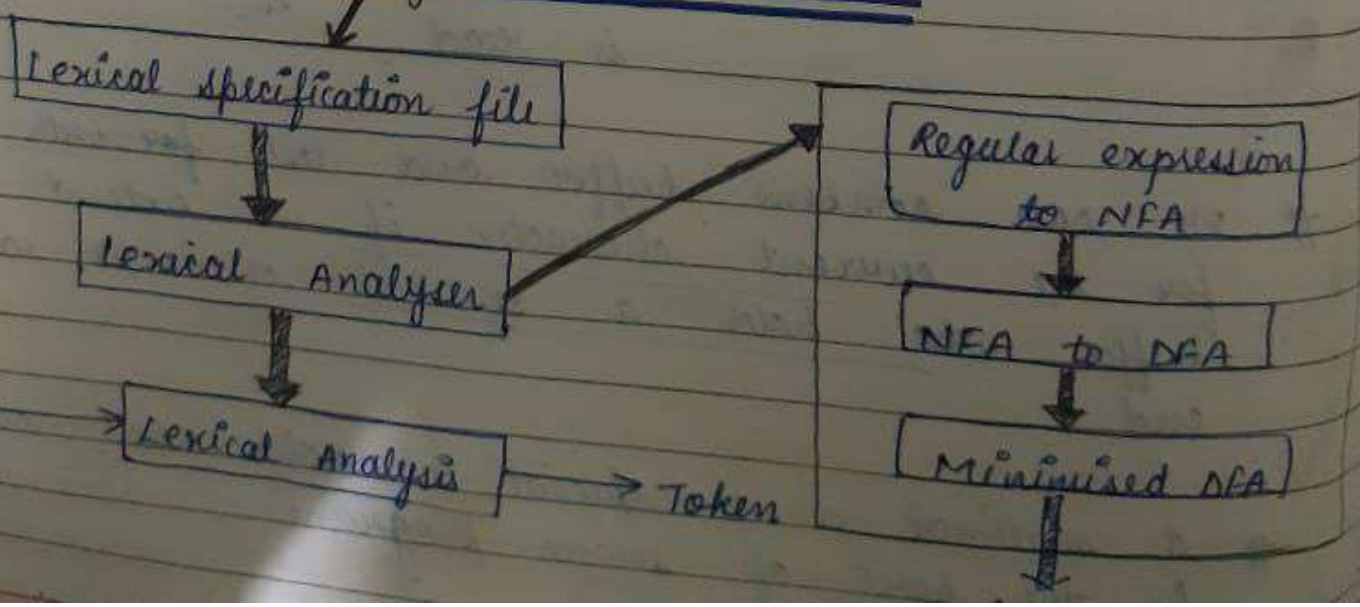
We can combine buffer end test for with the test for the current character if we extend each buffer who hold a sentinal character at the end

A sentinal is a special character that can not be the part of source program

Code to advance forward pointer.

```
fp=fp+1;
if (fp != EOF)
{
    if (fp is at end of first buffer)
    {
        Reload the data in second buffer;
        fp++;
    }
    else if (fp is at the end of 2nd buffer)
    {
        Reload the data in 1st buffer;
        fp = fp+1;
    }
    else
        fp = fp+1;
}
```

Regular Definition



2015 Page No. Date / /

Q: Write the regular definition of faxts number.

91 - (512) - 222 - 847
↓ country code
↓ area code
↓ exchange
↳ number.

digit → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
country → digit +
area → '(' digit + ')'
exchange → digit +
number → digit +

fax number → country '-' area '-' exchange '-' number

-ANS

Write the regular definitions of identifier.

alphabets → a | - - - - | z | A | - - - - | z.
symbols → $ | _
digits → 0 | - - - - - - | 9

start → alphabets | symbols | digits

identifier → start (alphabets + symbols + digits)*