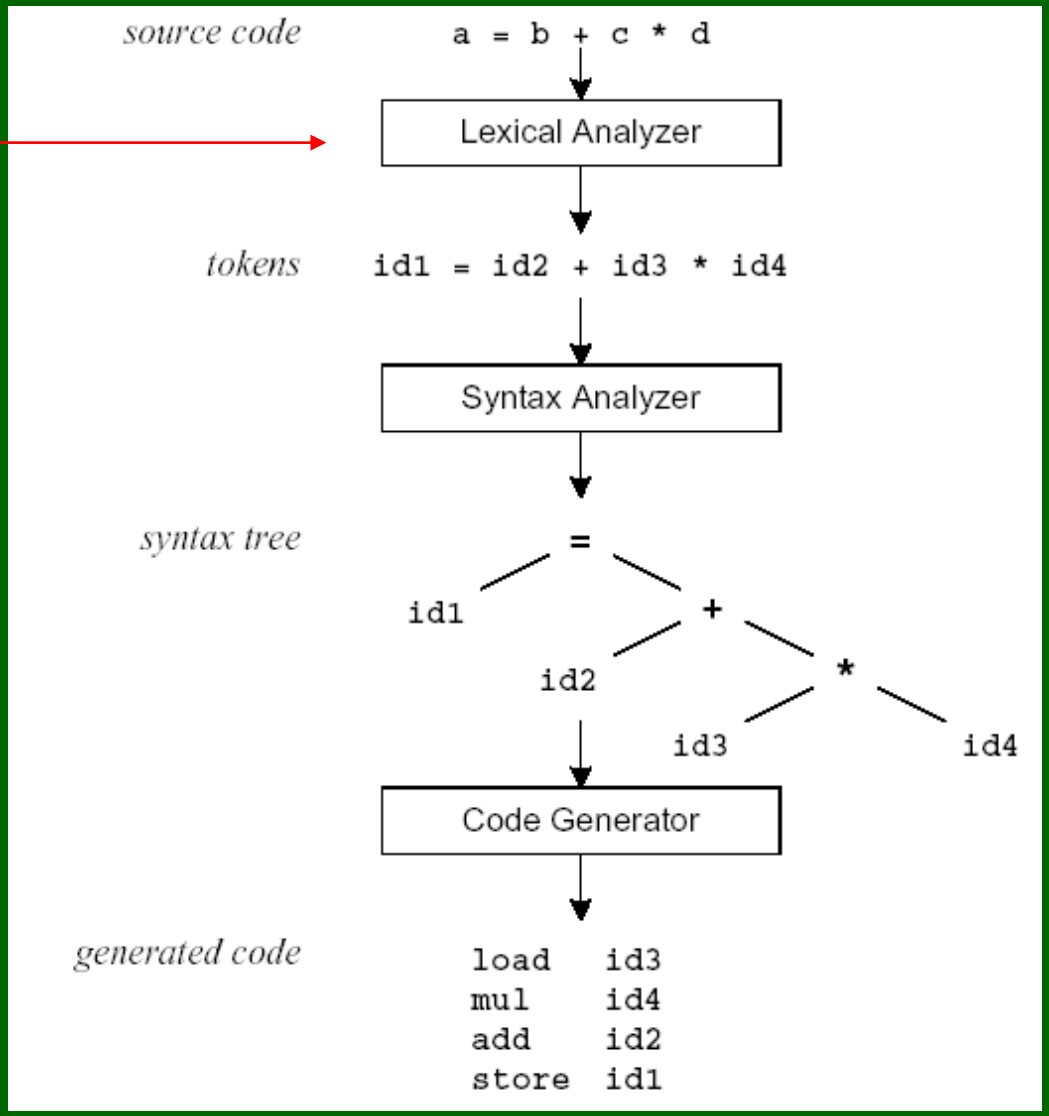
Genel olarak ve

Kısaca Lex!

Derleme Süreci



Lex?

Sözcüksel analiz edicinin (tarayıcı) ana görevi, bir girdi akışını daha kullanışlı öğelere (belirteçler) parçalamaktır.

```
a = b + c * d;
```

ID ASSIGN ID PLUS ID MULT ID SEMI

- Lex hızlı bir biçimde tarayıcı üretmek için kullanılabilecek bir araçtır.

Lex – Lexical Analyzer

- Lexical analyzer giriş katarını **token** lara dönüştürür.
- Token ise programlama dilinin **terminals** simgeleridir.
- Türkçe
  - kelime, sıfat, isim, zarf, yüklem, …
  - Programlama Dili
    - Identifiers, operators, keywords, …
- Regular ifadeler **terminals/tokens** tanımlar

Lex Source Program

- Lex source bir
  - regüler ifade ve buna uyan program fragman tablosudur.

```
digit  [0-9]
letter [a-zA-Z]
%%
{letter}({letter}|{digit})*       printf("id: %s\n", yytext);
\n                                printf("new line\n");
%%
main() {
        yylex();
}
```

Lex Source bir C Programı oluşturur.

- Tablo bir C programına çevrilir.(lex.yy.c) Bu program,
  - Giriş dizisini okur
  - girdiyi verilen ifadelerle eşleşen dizelere bölme ve
  - Gerekiyorsa onu çıkış dizisine kopyalama

LEX ve YACC içerikleri

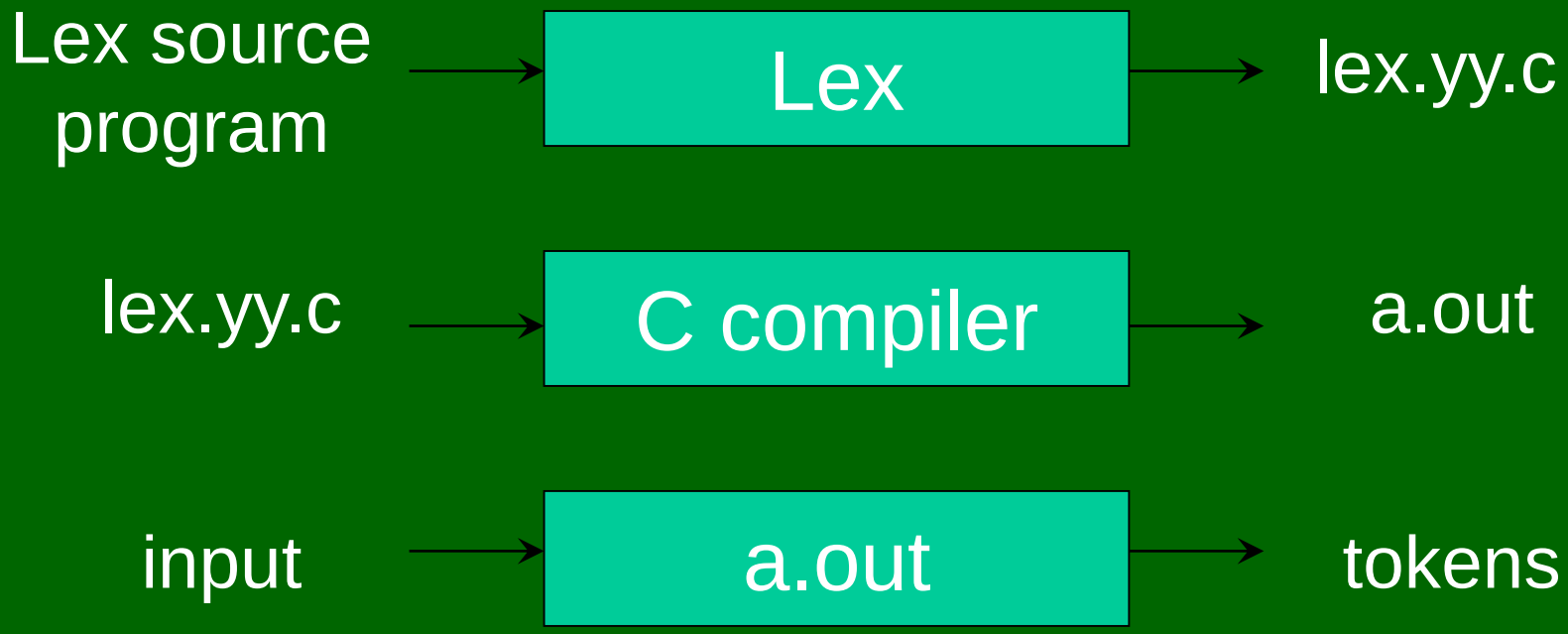
- LEX: Tarayıcı (scanner) üretmek için kullanılan bir programdır.
- Programcı regüler ifadelerin bir uzantısını kullanarak token olarak üretilen kelimeleri düzenler.
- LEX, yylex() adında bir C fonksiyonu oluşturur.
- LEX, YACC ile kullanılmak üzere tasarlanır.
- YACC, yyparser() isimli bir C fonksiyonu üreten bir parserdir.
- YACC, bir token'i okumak istediğinde yylex()'e çağrılar içeren bir parserdir.

Lex / Yacc

- Lex
  - Lex, bir lexical analyzer yada scanner için C kodu üretir.
  - Lex, girişteki katarla eşleşen desenleri kullanır ve katarı tokene dönüştürür.

- Yacc
  - Yacc, syntax analyzer yada parser için C kodu üretir
  - Yacc, LEX'in çıkışı olan tokenlerin analiz edilmesini sağlamak için gramer kurallarını kullanır ve bir syntax ağacı oluşturur.

Lex


Lex source program
Lex
lex.yy.c
lex.yy.c
C compiler
a.out
input
a.out
tokens

Lex Source

- Bir Lex source %% ile 3 kısma ayrılır: by delimiters
- Lex source dosyasının genel formatı:

```
{definitions}
%%                                  (required)
{transition rules}
%%
                                    (optional)
{user subroutines}
```

- Minimum Lex programı bu durumda

```
%%
```

Birinci Kısım

- LEX tanımları (MACRO) ve C bildrimlerinden oluşur.
- Bu LEX tanımları ikinci kısımdaki desenlerle kullanılır.
- MACRO desen içinde kullanıldığında { } ile kullanılır.
- Bu kısımdaki C bildirimleri yylex() fonksiyonu için global değişkendir.

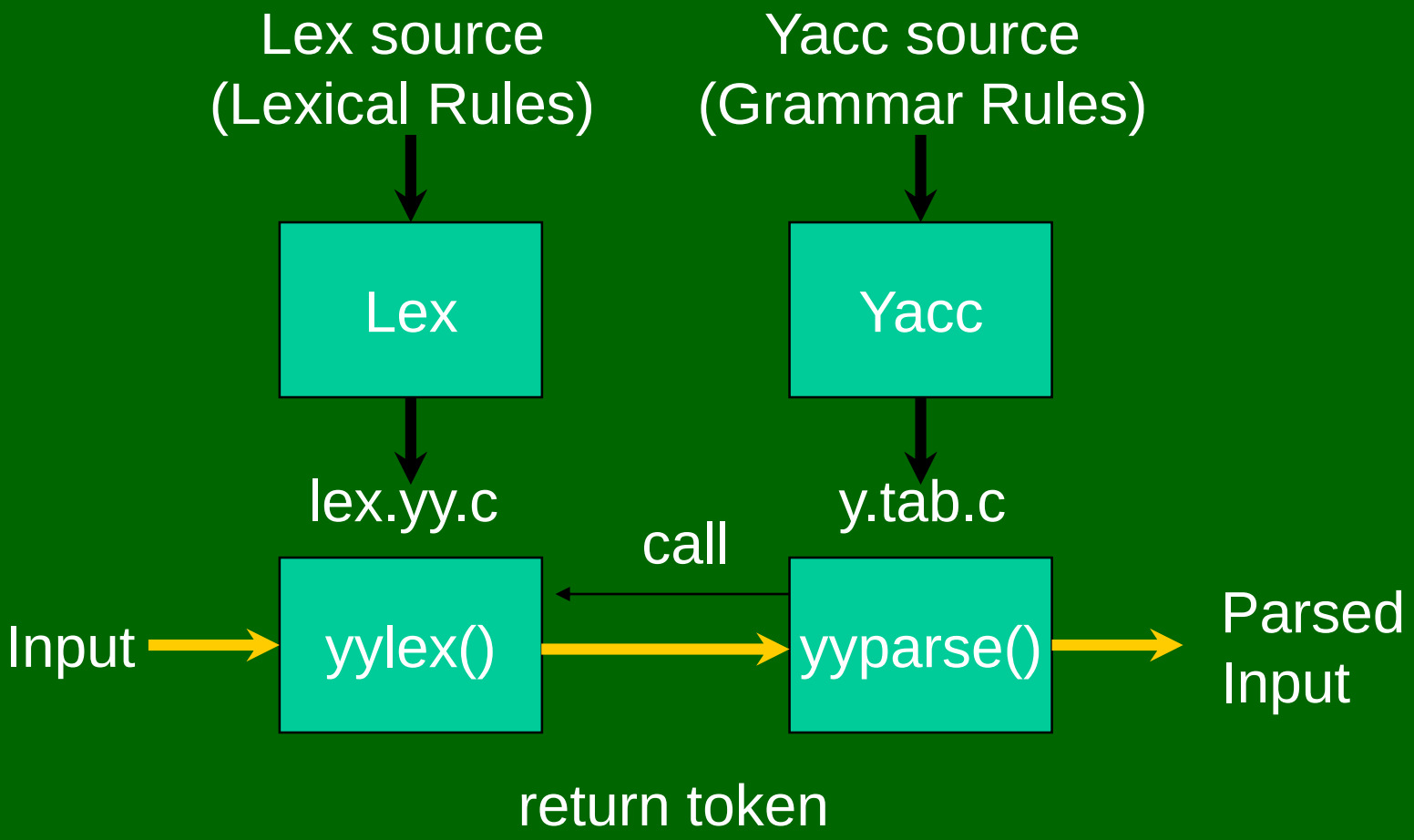
İkinci Kısım

- LEX desen ve aksiyonlarından oluşur.
- Lexical tokenleri tanımlayan kurallar kısmıdır.
- Her satırda bir desen ve bir aksiyon vardır.
- yylex(), desenlerden biri ile eşleşen bir girişi bulursa ilgili aksiyon çalışır.
- LEX, char *yytext ve int yylength değişkenlerini kullanıcının kullanımı için global olarak üretir.
- yytext:desen ile eşleşen giriş katarları
- yyleng: Bu katarın uzunluğu

Üçüncü Kısım

- Sadece C fonksiyonlarından oluşur.
- Bu fonksiyonlar hiçbir değişiklik yapılmaan lex çıktı dosyasına kopyalanır.
- LEX'in YACC ile kullanılma durumuna bağlı olarak main () fonksiyonu eklenmesine gereksinim duyulabilir.

Lex, Yacc ile bağımlı ise



Regüler İfadeler

Lex Regüler İfadeler

- Bir katar ile eşleşen regüler ifadeler:
- Regular ifade:
  - Operatörler
  - Karakter sınıfları
  - Keyfi karakterler
  - İsteğe bağlı ifadeler
  - Seçimlik ve Guruplandırma
  - Bağlam bağımlılık (Context sensitivity)
  - Tekrarlar ve tanımlar

  olarak sınıflandırılabilir.

Operatörler

" \ [ ] ^ - ? . * + | ( ) $ / { } % < >

- Metin karakterleri olarak kullanılacaklarsa, aşağıdaki kullanım geçerlidir.

  \$ = "$"

  \\ = "\"

- *blank*, *tab* (\t), *newline* (\n) haricindeki her karakter ve üstteki liste bir text karakteridir.

Karakter sınıfları [ ]

- `[abc]` bir karakterle (`a`, `b`, yada `c`) eşleşir.

`[ab]`	=> `a` yada `b`
`[a-z]`	=> `a` yada `b` yada `c` yada … yada `z`
`[-+0-9]`	=> bütün rakamlar ve iki işaret
`[^a-zA-Z]`	=> a dışında herhangi bir karekter

Keyfi karakterler

- (.):Enter dışındaki bütün karakterler sınıfı .
- `[\40-\176]` bütün basılabilen karakterler (octal 40 (blank) , 176(tilde~) arası)

İsteğe bağlı & Tekrarlı İfadeler

- `a?` => sıfır yada bir tane `a`
- `a*` => sıfır yada daha fazla `a`
- `a+` => bir yada daha fazla `a`

- Örnek:

  `ab?c` => `ac` veya `abc`

  `[a-z]+` => küçük harflerle bütün katarlar

  `[a-zA-Z][a-zA-Z0-9]*` => alfabetik başlayan bütün alfasayısal katarlar

Operatörlerin işlem önceliği

- Öncelik seviyeleri:
  - Kleene closure (*), ?, +
  - concatenation
  - alternation (|)
- Bütün operatörler soldan birleşmeli.

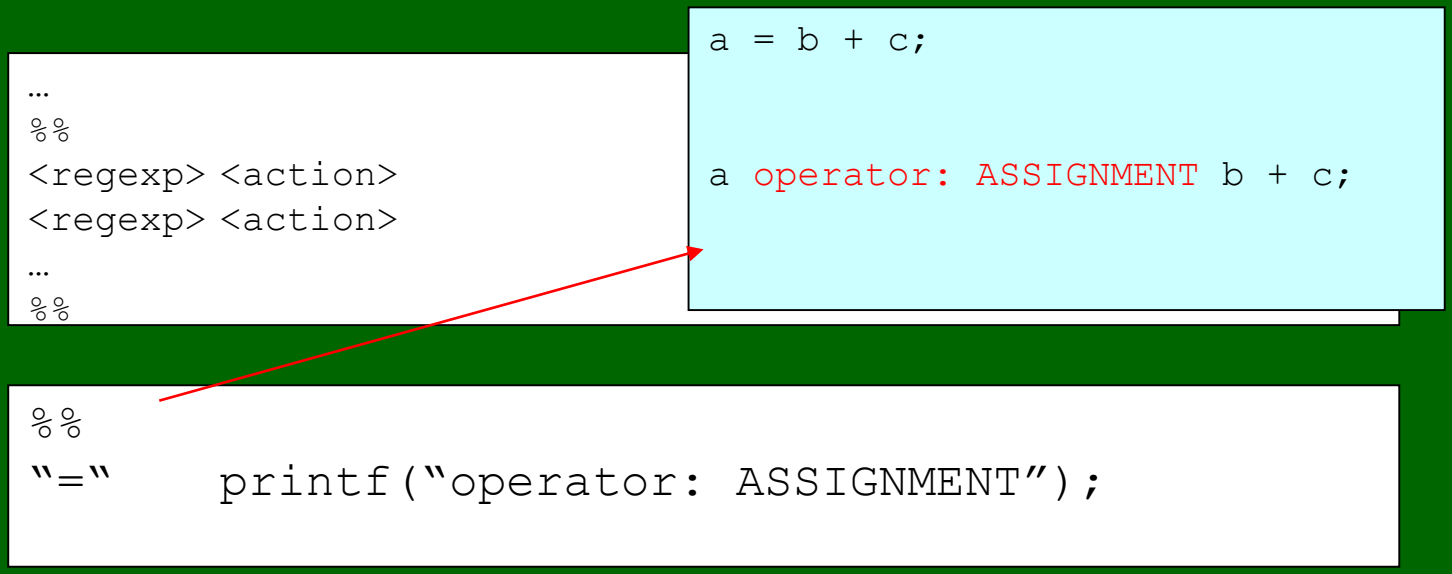
`a*b|cd*` = `((a*)b)|(c(d*))`

Desen Eşleştirme Temel Araçları

Metacharacter	Matches
`.`	any character except newline
`\n`	newline
`*`	zero or more copies of the preceding expression
`+`	one or more copies of the preceding expression
`?`	zero or one copy of the preceding expression
`^`	beginning of line / complement
`$`	end of line
`a|b`	`a` or `b`
`(ab)+`	one or more copies of `ab` (grouping)
`[ab]`	`a` or `b`
`a{3}`	3 instances of `a`
`"a+b"`	literal "`a+b`" (C escapes still work)

Recall: Lex Source

- Lex source bir regüler ifade ve buna uyan fragman tablosudur.



Geçiş Kuralları

- regexp <bir veya daha fazla boşluk> action (C code);
- regexp < bir veya daha fazla boşluk > { actions (C code) }

- Bir null statement ; girişte yok sayılacaktır. (no actions)

  `[ \t\n] ;`

  – Üç boşluk karakterinin yoksayılmasına neden olur.

```
a = b + c;
d = b * c;

↓↓

a=b+c;d=b*c;
```

- 4 özel seçenek:
  |, ECHO;, BEGIN, ve REJECT;
- | bu kuralın eyleminin bir sonraki kuralın eyleminden geldiğini gösterir

```
–  [ \t\n]    ;
–  " "         |
   "\t"        |
   "\n"        ;
```

- Eşleşmeyen simge, girişten çıkışa ECHO'nun varsayılan bir eylemi kullanıyor

- REJECT
  - Bir sonraki alternatifi yap

```
…
%%
pink        {npink++; REJECT;}
ink         {nink++; REJECT;}
pin         {npin++; REJECT;}
. |
\n          ;
%%
…
```

Lex öntanımlıDeğişkenler

- yytext -- lexeme lerden oluşan bir katar
- yyleng -- lexeme'nin uzunluğu
- yyin -- the input stream pointer
  - the default input of default main() is **stdin**
- yyout -- the output stream pointer
  - the default output of default main() is **stdout**.
- **cs20: %./a.out < inputfile > outfile**

- E.g.

```
[a-z]+          printf("%s", yytext);
[a-z]+          ECHO;
[a-zA-Z]+       {words++; chars += yyleng;}
```

Lex Kütüphane rutinleri

- yylex()
  - main() , yylex()'e çağrı içerir.
- yymore()
  - Bir sonraki tokeni döndür
- yyless(n)
  - yytext içindeki ilk n karakteri sakla
- yywarp()
  - Lex bir (end-of-file)'e ulaşınca çağrılır ve yywarp() 1 üretir (default olarak)

Lex öntanımlı değişkenler

Name	Function
`char *yytext`	pointer to matched string
`int yyleng`	length of matched string
`FILE *yyin`	input stream pointer
`FILE *yyout`	output stream pointer
`int yylex(void)`	call to invoke lexer, returns token
`char* yymore(void)`	return the next token
`int yyless(int n)`	retain the first n characters in yytext
`int yywrap(void)`	wrapup, return 1 if done, 0 if not done
`ECHO`	write matched string
`REJECT`	go to the next alternative rule
`INITAL`	initial start condition
`BEGIN`	condition switch start condition

Kullanıcı Altprogramları Bölümü

- LEX rutinlerini diğer programlama dillerinde kullandığınız biçimiyle kullanabilirsiniz.

```
%{
   void foo();
%}
letter       [a-zA-Z]
%%
{letter}+    foo();
%%
…
void foo() {
   …
}
```

Kullanıcı Altprogramları Bölümü

- main() yerleştiği kısım

```
%{
   int counter = 0;
%}
letter [a-zA-Z]

%%
{letter}+     {printf("a word\n"); counter++;}

%%
main() {
   yylex();
   printf("There are total %d words\n", counter);
}
```

Kullanım şekli

- Kaynak dosyadaki LEX'i çalıştırmak için;
  `lex scanner.l` yazılır.
- Bu lex.yy.c isimli bir dosya üretir. Bu ise lexical analiz edici için bir C programıdır.
- Bu lex.yy.c dosyasını derlemek için,
  `cc lex.yy.c -ll` yazılır.
- Lexical scanner programını çalıştırmak için
  `./a.out < inputfile` yazılır.

Lex sürümleri

- AT&T -- lex
  http://www.combo.org/lex_yacc_page/lex.html
- GNU -- flex
  http://www.gnu.org/manual/flex-2.5.4/flex.html
- a Win32 version of flex :
  http://www.monmouth.com/~wstreett/lex-yacc/lex-yacc.html
  or Cygwin :
  http://sources.redhat.com/cygwin/

- Lex farklı makinelerde aynı oluşturulmayabilir.