## Using States in Lex

- **Some regular languages are more easily expressed as FSA**
  - Set of all strings representing binary numbers divisible by 3
- **Lex allows you to use FSA concepts using *start states***
  - `%x MOD1 MOD2`
  - `"1"  {BEGIN MOD1}`
  - `"0"  { }`
  - `<MOD1> "1" {BEGIN 0}`



## Other Special Directives

- **ECHO causes Lex to echo current lexeme**
- **REJECT causes Lex to abandon current match and try an alternate one**
- **Example**

```
a|
ab|
abc|
abcd      {ECHO; REJECT;}
.|\n      {/* eat up the character */}
```



## Direct Construction of DFA from RE

- **Define notion of derivative of an RE *R* wrt a symbol *s***
  - ***R'*** such that ***sR'*** matches the exact same set of strings as ***R***
- incl_eps(R) = true, if R matches empty string
  false, otherwise
- Note:
  - **incl_eps(P|Q) = incl_eps(P) || incl_eps(Q)**
  - **Incl_eps(PQ) = incl_eps(P) && incl_eps(Q)**



- **D(a, a) = ε**
- **D(b, a) = ϕ**
- **D(P*, a) = D(P, a) P***
- **D(PQ, a) = D(P,a)Q | D(Q, a), if incl_eps(P)**
  **= D(P,a)Q otherwise**
- **D(P|Q, a) = D(P,a)|D(Q,a)**