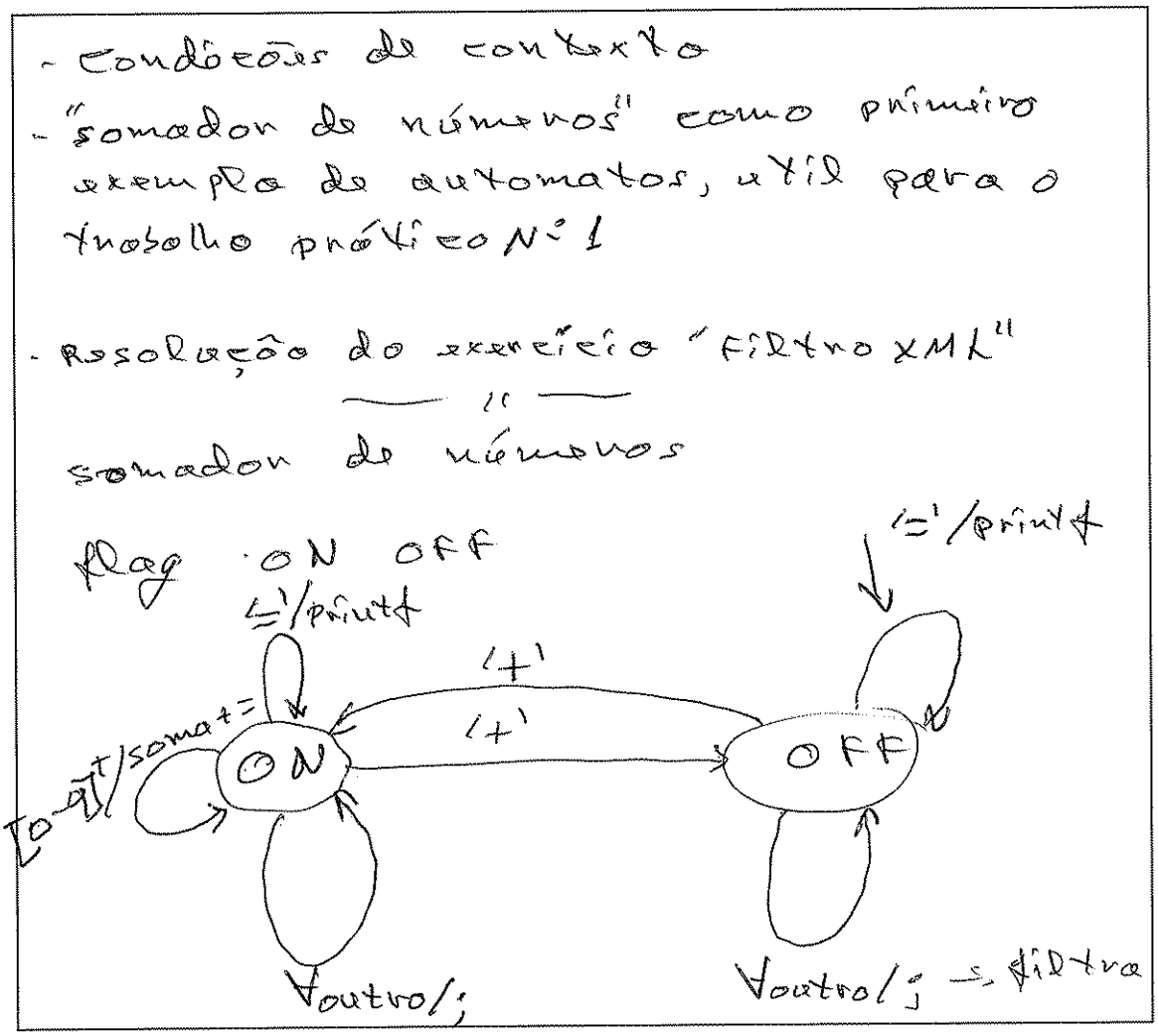
Universidade do Minho

Transcrição de aula

Disciplina	**Processamento de Linguagens – 3º ano - LEI**		
Secretário	**Número:** 7433	**Nome:** Delfim Macedo	
Data:	2011-03-24		**Nº Página**
Turno:	TP3		**Nº Alunos**

SUMÁRIO Autómatos

Análise Léxica



\<[^>]+\>

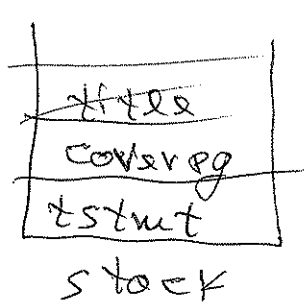
— " —

```
%{
  int x_open, x_close, x_empty;
%}
\<[^/]"/>"      x_empty++;
"</"[^>]\>      x_close++;
\<[^>]\>        x_open++;
%%
yywrap() {
        printf("\n Abertos = %d Fechados = %d Vazios = %d \n",
                x_open, x_close, x_empty++);
        return 0;
}
```

— " —

garantir que as TAGS) abram e fechem como devem ser.

Usando uma stack



stack

Análise Lexica, sintáxica

2ºm exemplo!

$T_{simais} = \{'+', '*', '=', '?', '!'\}$

$T_{palavrasreservadas} = \{\}$

$T_{variáveis} = \{id, num\}$

53 + 5

a = 7 * 2

?;

!a +;

tokens.h

```
#define MAIS    1000
#define MUL     1001
#define IGUAL   1002
#define !N      1003
#define OUT     1004
#define ID      3000
#define NUM     3001
```

flex

```
%{
  #include "tokens.h"
%}
%%
\+      return MAIS;
\*      return MUL;
...
```

```
[a-z]+      return ID;
[0-9]+      return NUM;
[\t\n]+     ;
.           return ERRO
```

tokenizer.c

```
#include "lex.yy.c"
#include <stdio.h>

int main(){
    int token;
    while((token = yylex()) != 0)
        printf("%d\n", token);
}
```