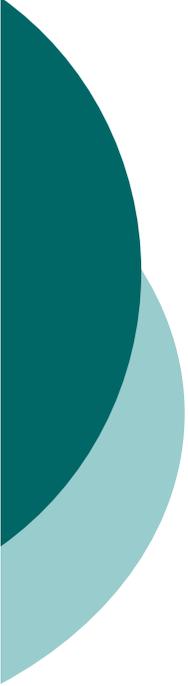


Universidade Regional de Blumenau

Implementação de Estrutura de Entrada e Saída de Dados para o Ambiente de Programação FURBOL

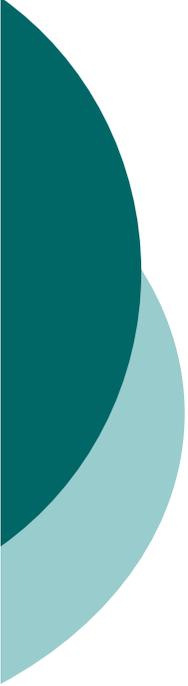
Aluno: Marcos Silva Piazero

Orientador: Antônio Carlos Tavares



Roteiro da Apresentação

- Introdução;
- Fundamentação Teórica;
- Projeto do sistema;
- Conclusões e Extensões.



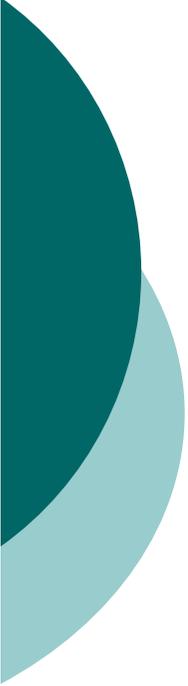
FURBOL

- Ambiente desenvolvido na Universidade Regional de Blumenau;
- Linguagem de Programação com vocabulário em Português.



Objetivos do Trabalho

- Implementar as rotinas de entrada e saída;
- Gerar chamadas no programa FURBOL às rotinas de entrada e saída.



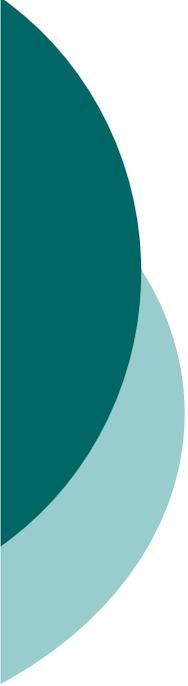
Fundamentação Teórica

- Ambientes de Programação;
- FURBOL;
- Sistemas Operacionais.



Ambientes de Programação

- Coleção de ferramentas para programação;
- Comandos que permitem ao desenvolvedor chamar funções do S.O.

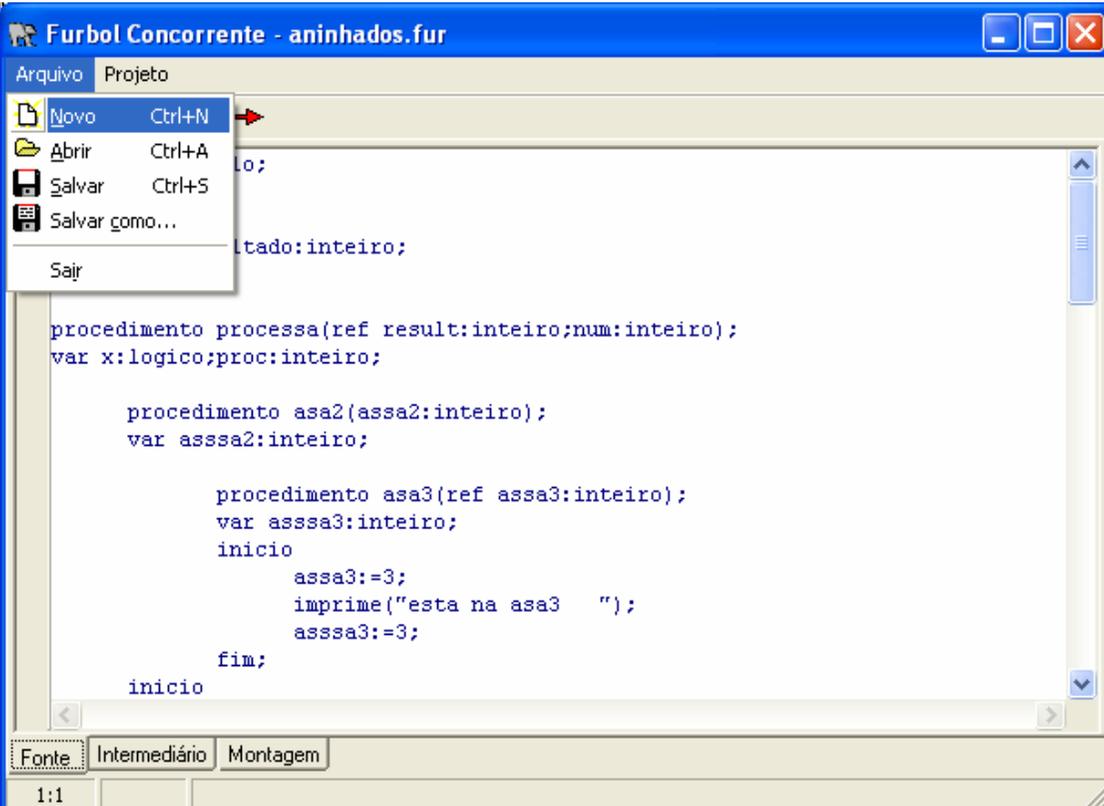


Compilador

- Traduz o programa de uma linguagem para um programa equivalente em outra linguagem.

FURBOL

- Edição dos programas



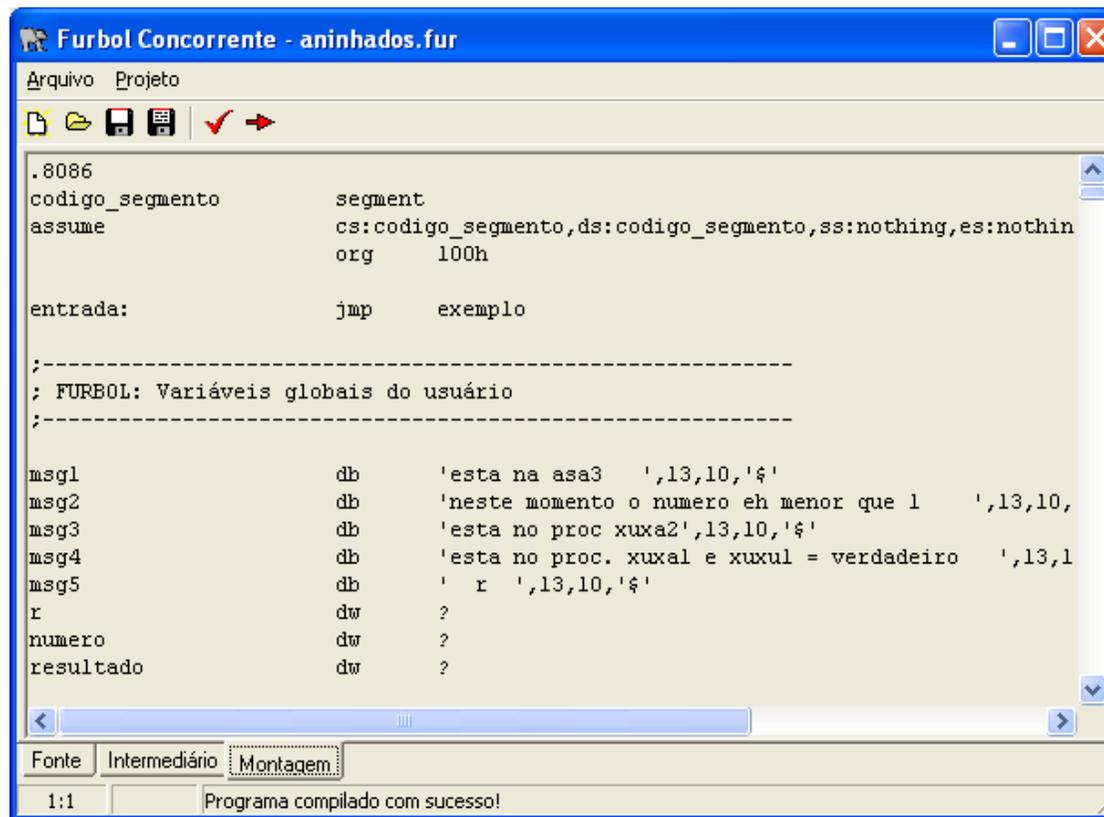
The screenshot shows the 'Furbol Concorrente - aninhados.fur' window. The 'Arquivo' menu is open, displaying options: 'Novo Ctrl+N', 'Abrir Ctrl+A', 'Salvar Ctrl+S', 'Salvar como...', and 'Sair'. The main editor area contains the following code:

```
lo;  
tado:inteiro;  
  
procedimento processa(ref result:inteiro;num:inteiro);  
var x:logico;proc:inteiro;  
  
    procedimento asa2(assa2:inteiro);  
    var assa2:inteiro;  
  
        procedimento asa3(ref assa3:inteiro);  
        var assa3:inteiro;  
        inicio  
            assa3:=3;  
            imprime("esta na asa3  ");  
            assa3:=3;  
        fim;  
    inicio
```

At the bottom, there are tabs for 'Fonte', 'Intermediário', and 'Montagem', with 'Fonte' selected. The zoom level is 1:1.

FURBOL

- Visualização Código *Assembly*



The screenshot shows a window titled "Furbol Concorrente - aninhados.fur". The window contains assembly code for an x86 program. The code includes segment definitions, a jump instruction, a comment for global variables, and several data definitions for messages and variables. The status bar at the bottom indicates "Programa compilado com sucesso!" (Program compiled successfully!).

```
.8086
codigo_segmento      segment
assume               cs:codigo_segmento,ds:codigo_segmento,ss:nothing,es:nothin
                    org      100h

entrada:             jmp      exemplo

;-----
; FURBOL: Variáveis globais do usuário
;-----

msg1                  db      'esta na asa3 ',13,10,'$'
msg2                  db      'neste momento o numero eh menor que 1 ',13,10,
msg3                  db      'esta no proc xuxa2',13,10,'$'
msg4                  db      'esta no proc. xuxal e xuxul = verdadeiro ',13,1
msg5                  db      ' r ',13,10,'$'
r                     dw      ?
numero                dw      ?
resultado             dw      ?
```

Fonte | Intermediário | **Montagem**

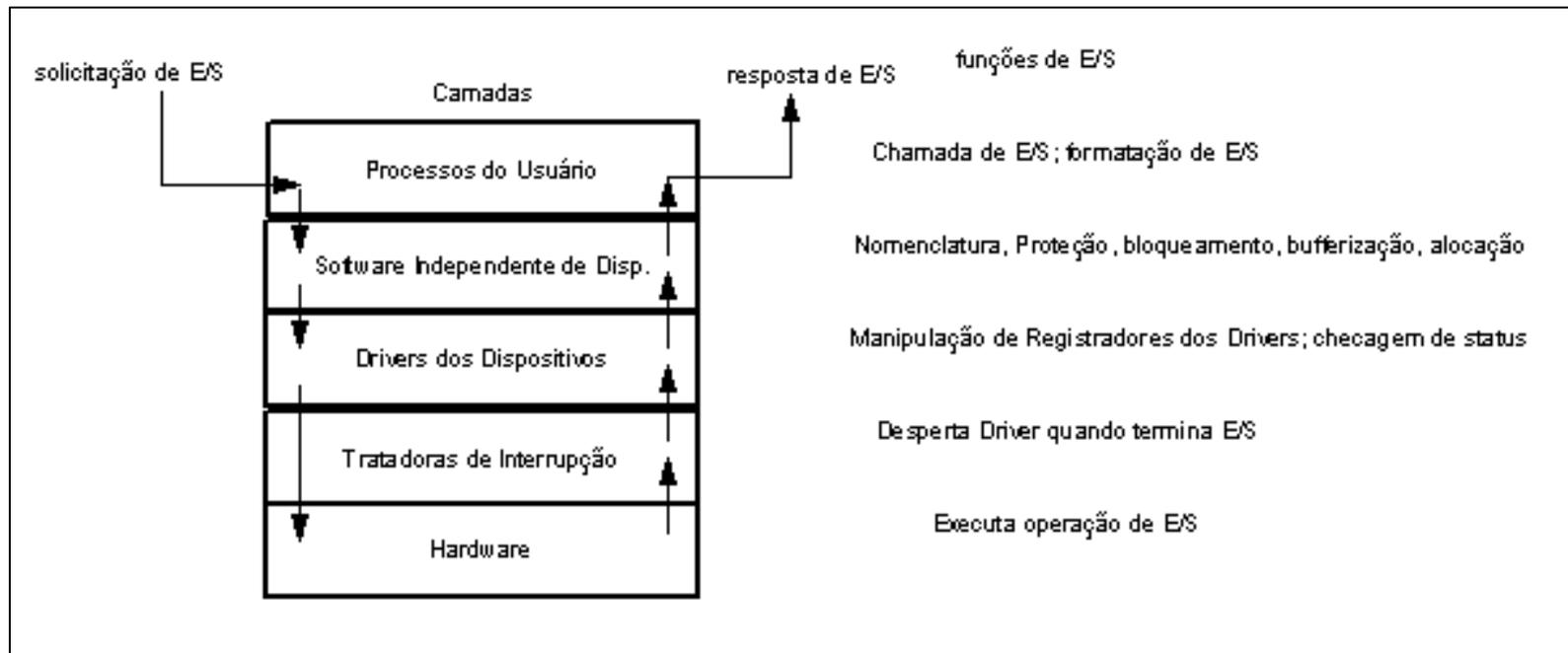
1:1 | Programa compilado com sucesso!

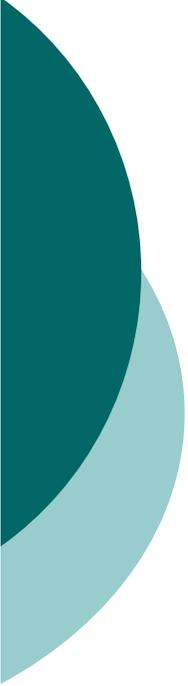


Sistemas Operacionais

- Gerencia recursos computacionais;
- Disponibiliza funções comuns de acesso aos recursos (Entrada e Saída de Dados por exemplo).

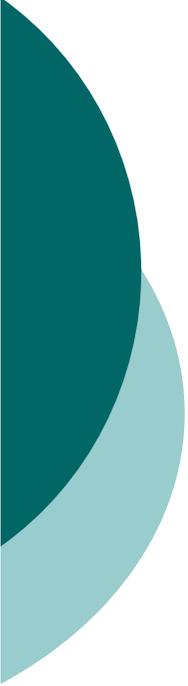
Software de E/S





Projeto do Sistema

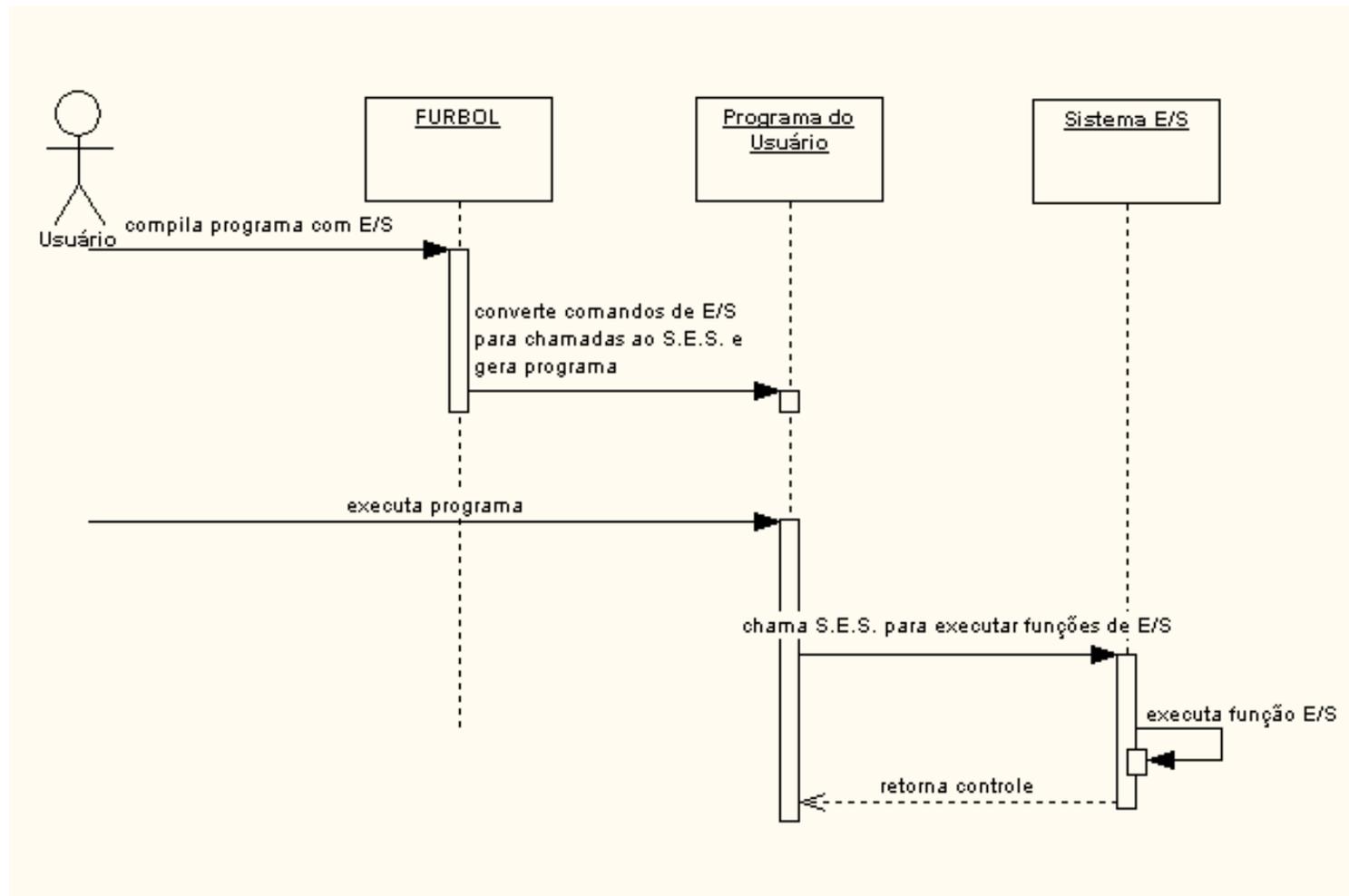
- Especificação;
- Implementação;
- Resultados.

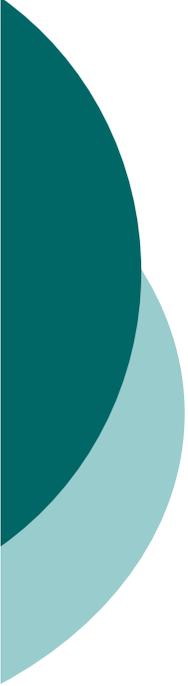


Requisitos

- Implementar funções de ES;
- Manter idioma de nomenclatura de comandos (Português).

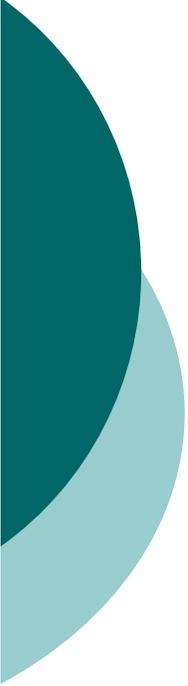
Diagrama de Seqüência





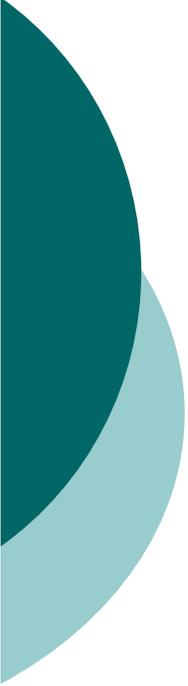
Novos Comandos

- crie;
- abra;
- feche;
- grave;
- leia.



Chamadas ao Sistema de ES

```
mov flag_es,1  
push parâmetros  
push PID  
push código função  
int 60h  
pop retorno  
pop PID  
pop parâmetros  
mov flag_es, 0
```



Sintaxe Comando 'crie'

Retorno := `crie(nome)`

nome: constante literal

Retorno: inteiro

- 0 – a criação falhou;
- >0 – *handle* do arquivo.



Ação Semântica comando 'crie'

```
mov flag_es, 1  
push endereço nome arquivo  
push PID  
push 1  
int 60h  
pop AX  
mov retorno, AX  
pop PID  
pop endereço nome arquivo  
mov flag_es, 0
```

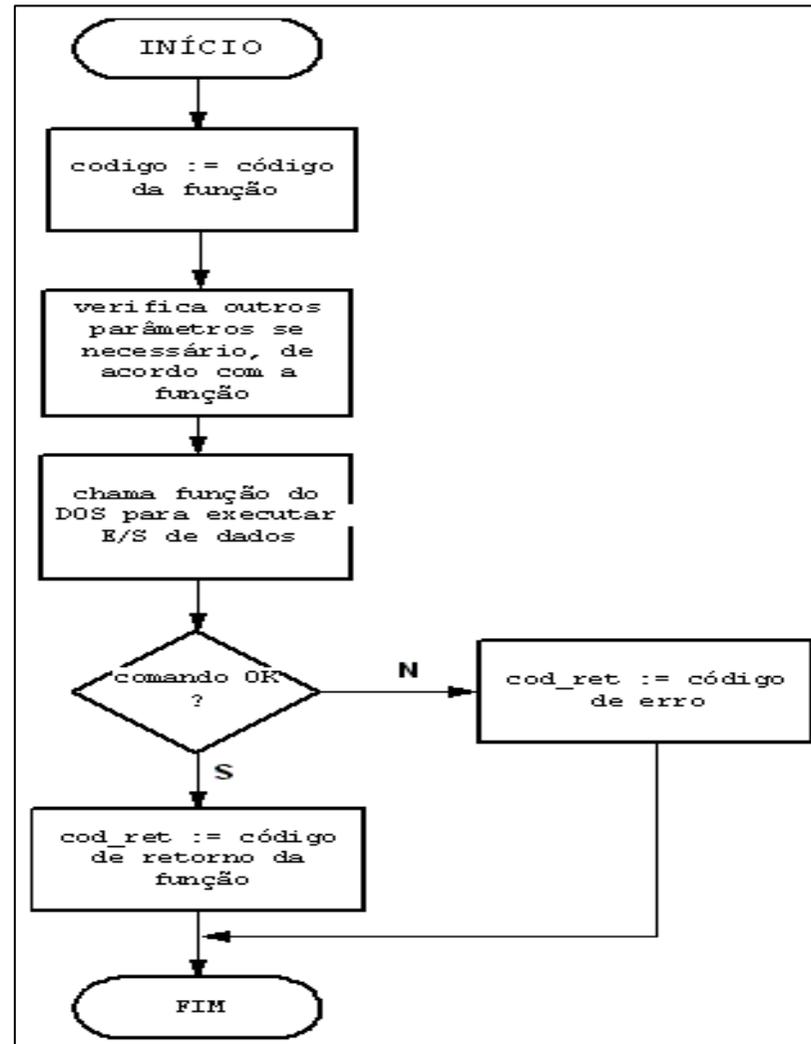
Alterações BNF

<i>CCrie</i>	→	'crie'	
		('	
		<i>ConstanteLiteral</i>	Nome := EmiteDados(<i>ConstanteLiteral</i>);
)'	<i>CCrie.codasm</i> := 'mov AX' 'offset' Nome CRLF; <i>CCrie.codasm</i> := <i>CCrie.codasm</i> + 'push AX' CRLF; <i>CCrie.codasm</i> := <i>CCrie.codasm</i> + 'mov AX,1' CRLF; <i>CCrie.codasm</i> := <i>CCrie.codasm</i> + 'push AX' CRLF; <i>CCrie.codasm</i> := <i>CCrie.codasm</i> + 'int 60' CRLF; <i>CCrie.codasm</i> := <i>CCrie.codasm</i> + 'pop AX' CRLF; <i>CCrie.codasm</i> := <i>CCrie.codasm</i> + 'pop AX' CRLF;

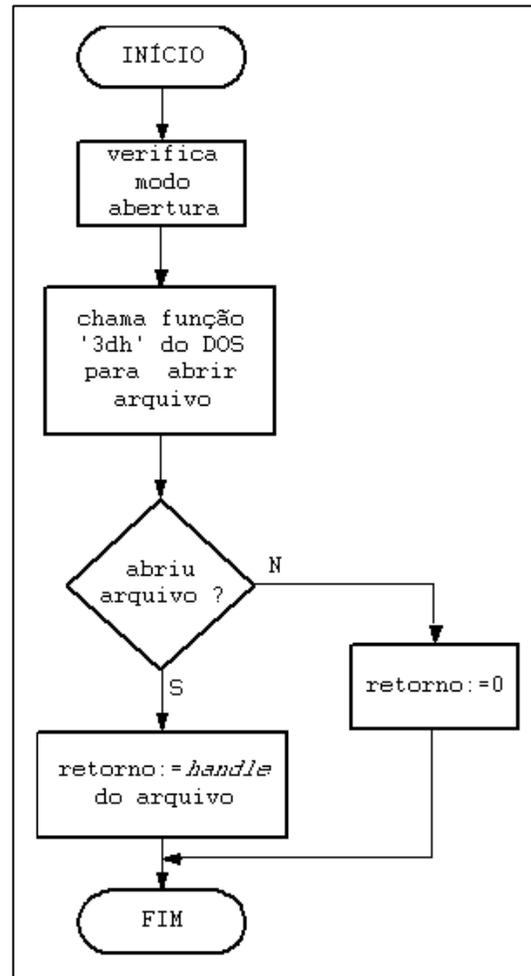
Alterações BNF

<i>CAbra</i>	→	'abra'	
		'('	
		ConstanteLiteral	Nome := EmiteDados(ConstanteLiteral);
		','	
		<i>Modo</i>	
)'	<i>CAbra.codasm</i> := 'mov AX' 'offset' Nome CRLF; <i>CAbra.codasm</i> := <i>CAbra.codasm</i> + 'push AX' CRLF; <i>CAbra.codasm</i> := <i>CAbra.codasm</i> + 'mov AX' ',' <i>Modo</i> CRLF; <i>CAbra.codasm</i> := <i>CAbra.codasm</i> + 'push AX' CRLF; <i>CAbra.codasm</i> := <i>CAbra.codasm</i> + 'mov AX,2' CRLF; <i>CAbra.codasm</i> := <i>CAbra.codasm</i> + 'push AX' CRLF; <i>CAbra.codasm</i> := <i>CAbra.codasm</i> + 'int 60' CRLF; <i>CAbra.codasm</i> := <i>CAbra.codasm</i> + 'pop AX' CRLF; <i>CAbra.codasm</i> := <i>CAbra.codasm</i> + 'pop AX' CRLF; <i>CAbra.codasm</i> := <i>CAbra.codasm</i> + 'pop AX' CRLF;

Funcionamento do Sistema



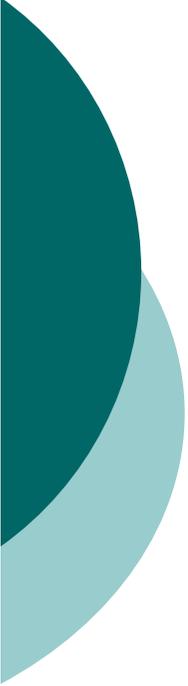
Funcionamento da função “abra”





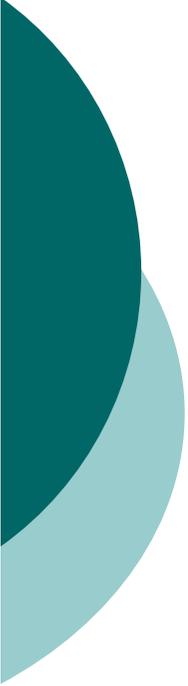
Implementação - Ferramentas utilizadas

- Ambiente Borland Delphi 6;
- Turbo Borland Pascal 7.0
(Linguagem Pascal e *Assembly*).



Alterações no FURBOL

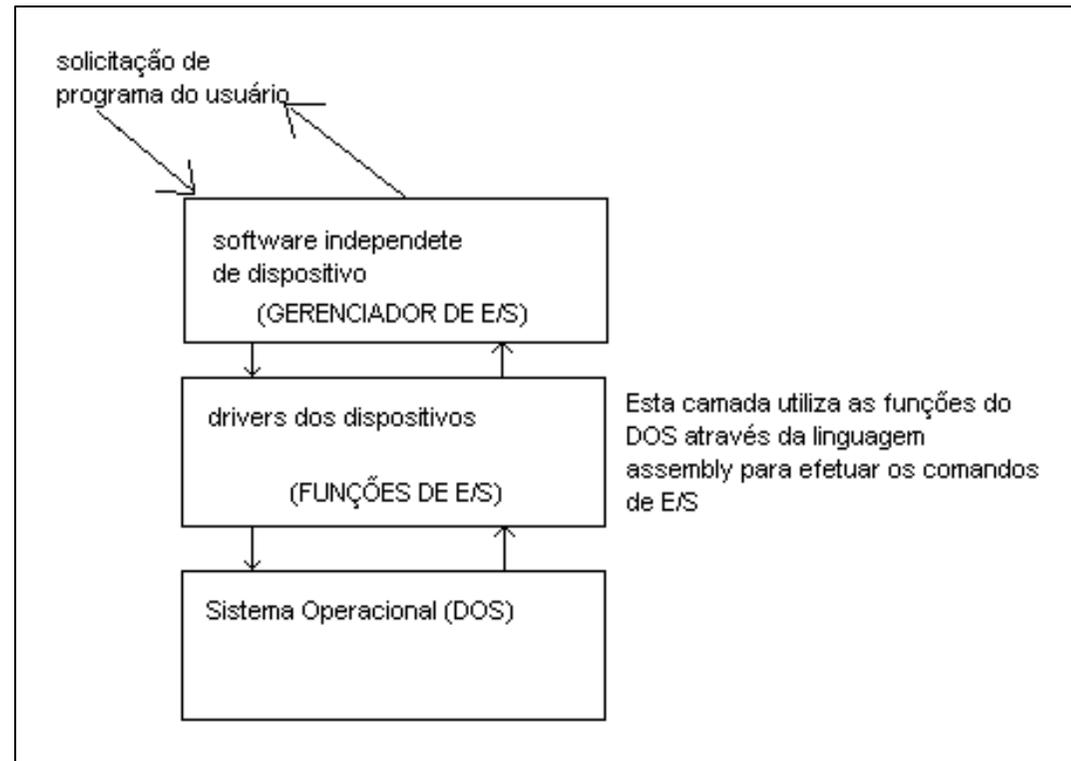
- Inclusão de novos comandos;
- Ações Semânticas;
- Comandos de E/S são realizadas de forma atômica;
- Alterada rotina do Escalonador de processos.

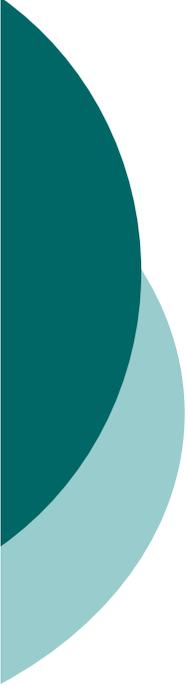


Sistema de ES

- Sistema residente em Memória;
- Utiliza funções do DOS;
- Independente do FURBOL.

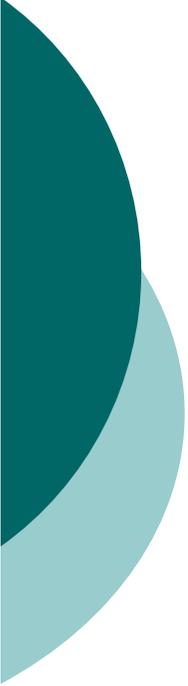
Arquitetura do Sistema de ES





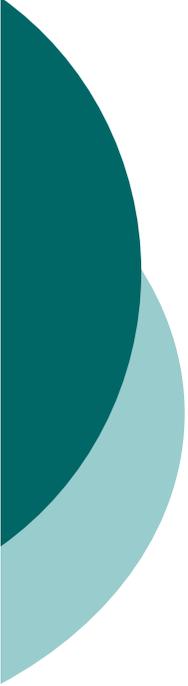
Resultados

- Criação e gravação;
- Abertura e leitura;
- Fechamento;
- Tipo de dado inteiro;
- Nome do arquivo é constante literal;
- Gravação só é possível após criação de arquivo.



Conclusões

- Arquitetura de camadas se mostrou simples;
- Tratamento de dados exigiu esforço;
- Número de funções de ES limitado;
- Independência (FURBOL – Sistema).



Extensões

- Novas funções;
- Desvincular sistema do DOS;
- Tornar Sistema de ES concorrente;
- Criar tipo de dados cadeia de caracteres no FURBOL.
- Adaptar sistema para outro ambiente de programação.