

Protótipo de Gerador de Código Executável no Ambiente FURBOL

Acadêmico: Geovânio Batista André

Orientador: José Roque Voltolini da Silva

Trabalho de Conclusão de Curso

Área/SubÁrea

Compiladores/Geração de Código

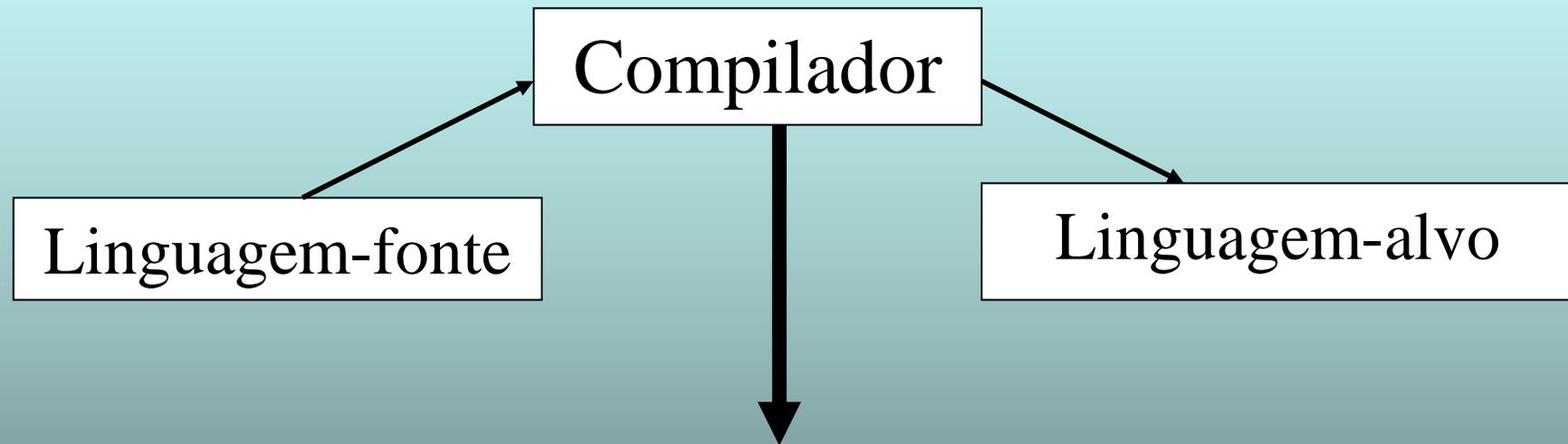
Roteiro

- Introdução
- Fundamentação Teórica
- Desenvolvimento
 - Especificação
 - Geração do Código Intermediário
 - Geração do Código Assembly
- Conclusão

Introdução

- FURBOL
 - Douglas (PIPe/TCC)
 - Joilson
 - Bruxel
 - Marcelo Radlof
 - Heldio Shimtz
- Objetivo

Compiladores



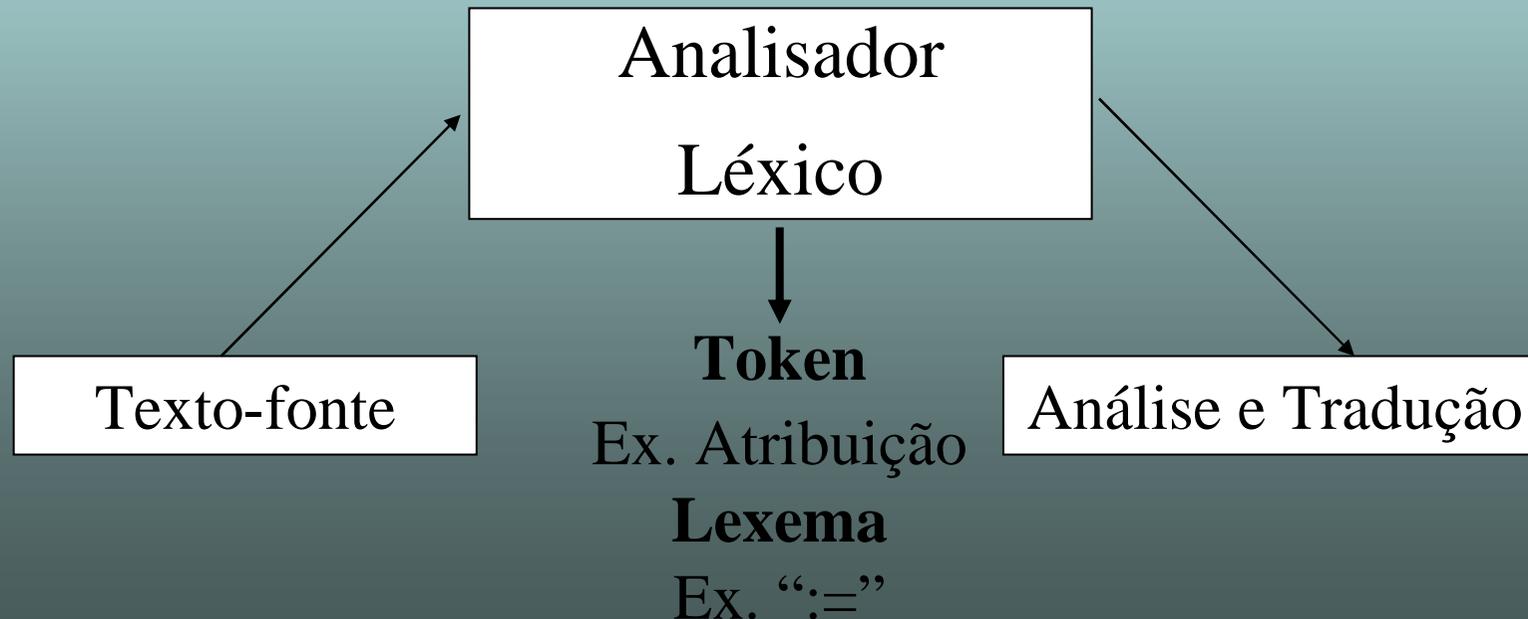
Análise Léxica

Análise Sintática

Análise Semântica

Análise Léxica

- Faz a interface entre o texto-fonte e os programas encarregados de sua análise e tradução



Análise Sintática

- Gramáticas Livre de Contexto
 - BNF (*Bakus Normal Form*)
- Análise Sintática Top-Down

Análise Semântica

- Definição dirigida por sintaxe
 - Gramática de Atributos
 - Atributos Herdados
 - Atributos Sintetizados

$D ::= TL$	$L.in := T.tipo$
$T ::= \mathbf{int}$	$T.tipo := inteiro$
$T ::= \mathbf{real}$	$T.tipo := real$
$L ::= L_1, \mathbf{id}$	$L_1.in := L.in$
	$incluir_tipo(\mathbf{id}.entrada, L.in)$
$L ::= \mathbf{id}$	$Incluir_tipo(\mathbf{id}.entrada, L.in)$

Análise Semântica

- Tabelas de símbolos;
- Identificadores;
- Tipos de dados;
- Escopo;
- Tradução do programa;
- Geração de código.

Código Intermediário

- Por que gerar código intermediário????
 - Diferentes máquinas-alvo
 - Otimizações independentes de máquina
 - Diminuição da complexidade na tradução

Código Intermediário

- Enunciados de três endereços

$$X := y \text{ op } z$$

SE NUM = NUM2 goto L2

Geração de Código

- Programa-alvo
- Gerenciamento de memória
 - *var x:integer;* (definição de variável em Pascal)
- Seleção de instruções
- Alocação de registradores
- Máquina-alvo - 8088 e compatíveis

Geração de Código

- Máquina-alvo (8088)
 - Registradores
 - Propósito geral (AX, BX, CX e DX)
 - Ponteiro e Índice (BP, SP, SI e DI)
 - Segmento (CS, DS, SS e ES)
 - Ponteiro da Instrução (IP)
 - Sinalizadores

Linguagem Assembly

[nome:] [operação] [operandos] [;comentário]

Turbo Assembler 1.01

ENTRADA: JMP TESTE ;salto incondicional

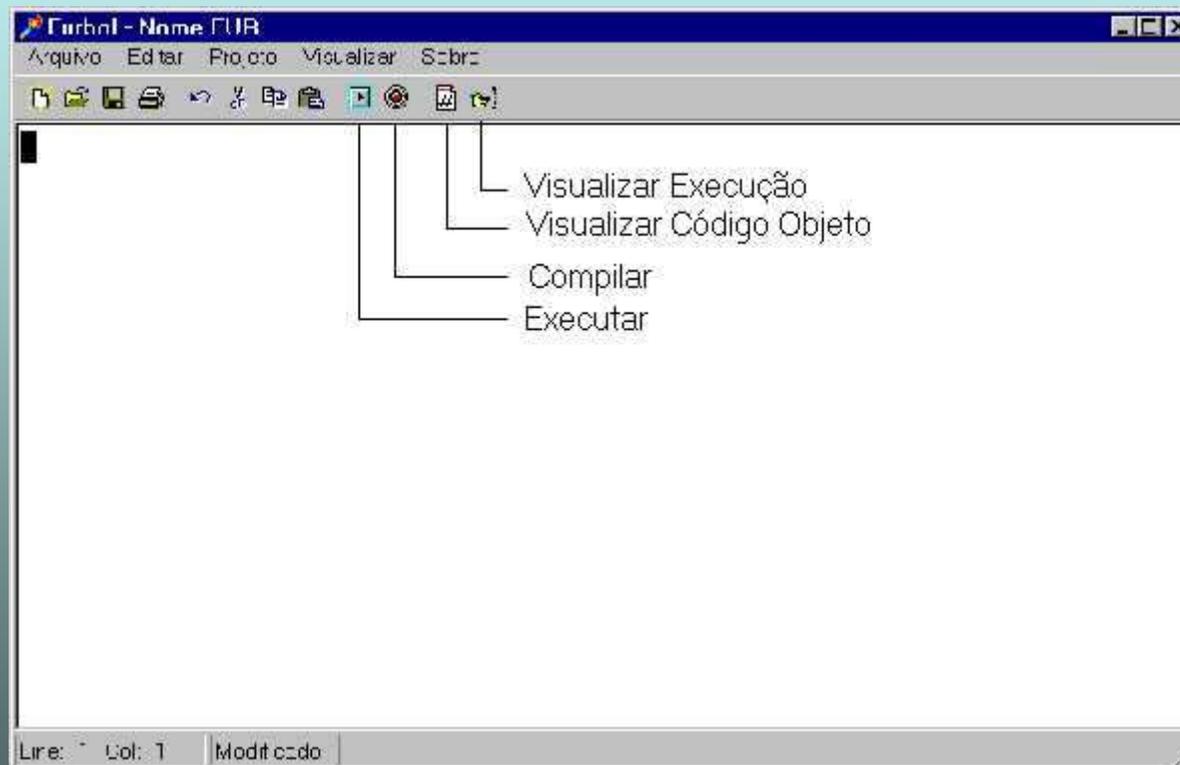
Linguagem Assembly

```
CODIGO          SEGMENT
  ASSUME CS:CODIGO, SS:CODIGO, DS:CODIGO, ES:CODIGO
  ORG           100H
INICIO:         JMP     COMECO
  |
  | definição de variáveis
  |
COMECO          PROC   NEAR
  |
  | corpo de instruções
  |
  INT 20H
COMECO          ENDP
CODIGO          ENDS
END             INICIO
```

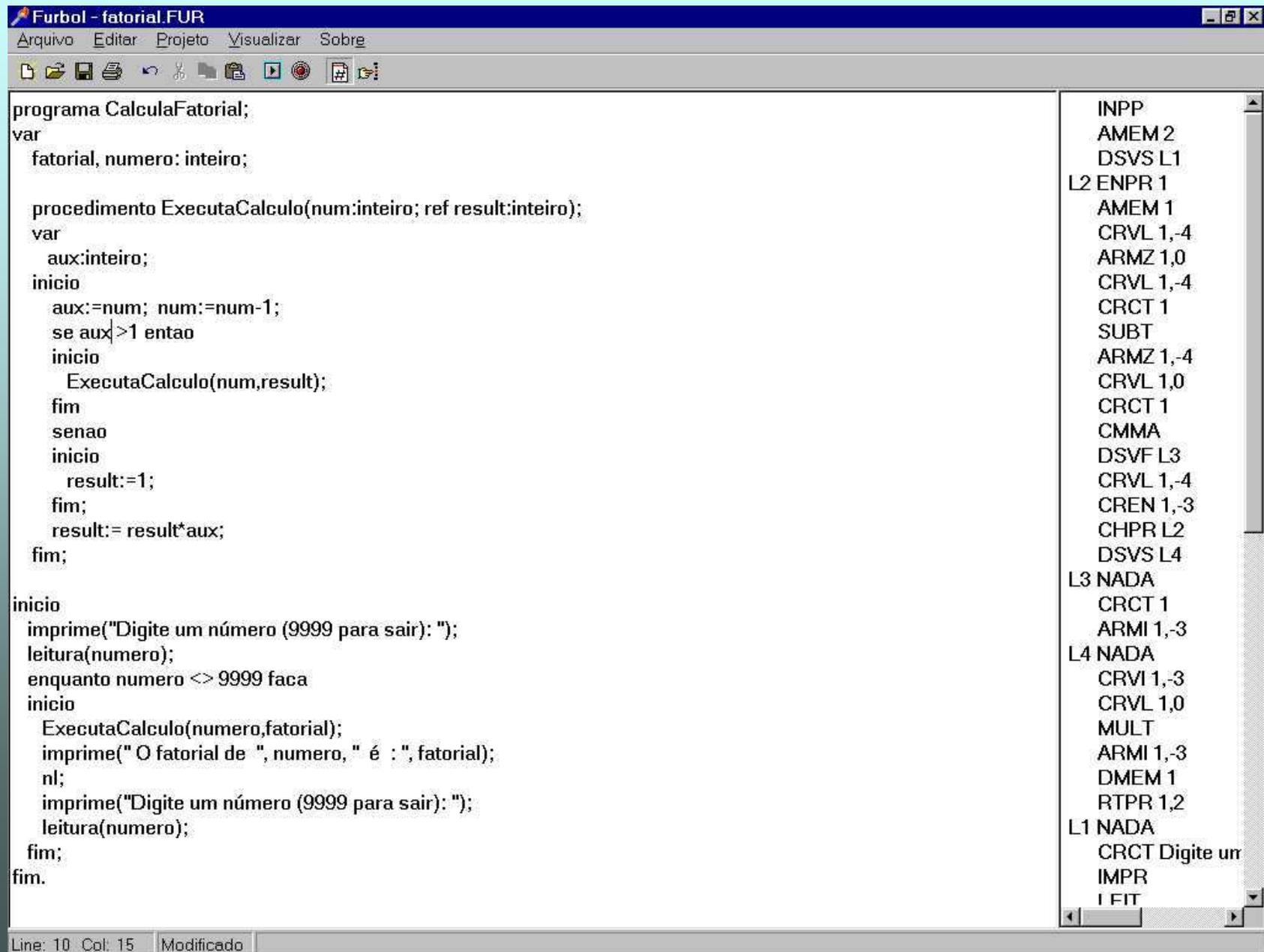
Ambiente FURBOL

- Dados *INTEIRO*, *LÓGICO* e *REGISTRO* ;
- Comando Condicional *SE ENTÃO SENÃO*;
- Comando de Repetição *ENQUANTO FAÇA*;
- Comandos de saída *IMPRIME* e comando de entrada *LEITURA*.
- Procedimento com passagem de parâmetros por cópia-valor e por referência (*REF*);
- Comando nova linha *NL*.
- Ambiente de Programação
 - Delphi 3.0

Tela Principal



Linguagem Mepa



The screenshot shows a window titled "FurboI - fatorial.FUR" with a menu bar (Arquivo, Editar, Projeto, Visualizar, Sobre) and a toolbar. The main area is split into two panes. The left pane contains Mepa source code for a factorial program. The right pane shows the corresponding assembly code, including instructions like INPP, AMEM, DSVS, ENPR, CRVL, ARMZ, CRCT, SUBT, CMMA, DSVF, CREN, CHPR, and NADA, along with labels L1, L2, L3, and L4. The status bar at the bottom indicates "Line: 10 Col: 15 Modificado".

```
programa CalculaFatorial;
var
  fatorial, numero: inteiro;

procedimento ExecutaCalculo(num:inteiro; ref result:inteiro);
var
  aux:inteiro;
inicio
  aux:=num; num:=num-1;
  se aux>1 entao
  inicio
    ExecutaCalculo(num,result);
  fim
  senao
  inicio
    result:=1;
  fim;
  result:= result*aux;
fim;

inicio
  imprime("Digite um número (9999 para sair): ");
  leitura(numero);
  enquanto numero <> 9999 faça
  inicio
    ExecutaCalculo(numero,fatorial);
    imprime(" O fatorial de ", numero, " é : ", fatorial);
    nl;
    imprime("Digite um número (9999 para sair): ");
    leitura(numero);
  fim;
fim.
```

INPP
AMEM 2
DSVS L1
L2 ENPR 1
AMEM 1
CRVL 1,-4
ARMZ 1,0
CRVL 1,-4
CRCT 1
SUBT
ARMZ 1,-4
CRVL 1,0
CRCT 1
CMMA
DSVF L3
CRVL 1,-4
CREN 1,-3
CHPR L2
DSVS L4
L3 NADA
CRCT 1
ARMI 1,-3
L4 NADA
CRVI 1,-3
CRVL 1,0
MULT
ARMI 1,-3
DMEM 1
RTPR 1,2
L1 NADA
CRCT Digite um
IMPR
I FIT

Line: 10 Col: 15 Modificado

Desenvolvimento do Protótipo

- Ambiente de Desenvolvimento
 - Delphi 5

Especificação

- Programas e Blocos
- Estrutura de Dados
- Estrutura de SubRotinas
- Estrutura de Comandos
- Estrutura de Controle de Expressões

Estrutura de SubRotinas

EstruturaProcedimento	::=	'PROCEDIMENTO'	
	,		
	#ID,		se encontra_var(id.nome)<>nil entao erro; T:=CriarTabela(Topo(PtrTab)); instalar_Proc(Topo(PtrTab),Id.Nome, t, Desvio); Empilhar(T,PtrTab);Empilhar(0,Deslocamento); Nret:=0;
	ParamFormais,		CComposto.Nret:=ParaFormais.Nret;
	',';		
	EstruturaDados, EstruturaSubRotinas		nivel:=nivel+1; nivel:=nivel-1;
	CComposto,		EstruturaProcedimento.CodigoAsm := ID.nome PROC NEAR PUSH BP MOV BP,SP SUB SP,Topo(deslocamento) CComposto.CodAsm MOV SP,BP POP BP RET Nret ID.nome ENDP EstruturaSuRotinas.CodAsm;
	',';		T:=Topo(PtrTab); Registrar_Largura(t,Topo(Deslocamento)); Desempilhar(PtrTab);Desempilhar(Deslocamento);

Código fonte \Rightarrow Código intermediário

```

ENQUANTO
NUM1 < 20 FACA

INICIO
NUM2 := NUM2 * NUM1 ;
INC ( NUM1 ) ;
FIM ;
    
```

```

L2 :
SE NUM1 < 20
goto L3
goto L1
L3 :
NUM2 := NUM2 * NUM1
INC ( NUM1 )
goto L2
L1 :
    
```

CRepetição	::=	'ENQUANTO',	CRepetição.início:=novo_L; Expressão.v:=novo_L; Expressão.f:=CRepetição.próx;
		Expressao,	
		'FACA',	
		ComandoComposto;	CComposto.prox:=CRepetição.início; CRepetição.código:= CRepetição.início ':' Expressão.código E.v ':' CComposto.código GOTO CRepetição.início; CRepetição.códiAsm:= CRepetição.início ':' Expressão.códAsm E.v ':' CComposto.códAsm JMP CRepetição.início;

Código intermediário ⇒ Código Assembly

```

L2 :
SE NUM1 < 20
goto L3
goto L1
L3:
NUM2 := NUM2 * NUM1
INC ( NUM1 )
goto L2
L1:
    
```

```

L2 :
MOV    AX , NUM1
CMP    AX , 20
JB     L3
JMP    L1
L3:
MOV    AX , NUM2
MUL    NUM1
MOV    NUM2 , AX
INC    NUM1
JMP   L2
L1:
    
```

CRepetição	::=	'ENQUANTO',	CRepetição.início:=novo_L; Expressão.v:=novo_L; Expressão.f:=CRepetição.próx;
		Expressao,	
		'FACA',	
		ComandoComposto;	CComposto.prox:=CRepetição.início; CRepetição.código:= CRepetição.início ':' Expressão.código E.v ':' CComposto.código GOTO CRepetição.início; CRepetição.códiAsm:= CRepetição.início ':' Expressão.códAsm E.v ':' CComposto.códAsm JMP CRepetição.início;

Conclusão

- Código executável
- Definição
- Arquivo executável (*.COM*)
- Limitações
 - Números negativos
 - Tipo registro
 - Nomes temporários de procedimentos envolventes
 - Comandos de I/O

Extensões

- Mapeamento finito (*array*).
- Otimização do código *assembly*.
- Formato *.EXE*.
- Disponibilidade de memória durante a execução do código gerado pelo ambiente.