

Construção de Compiladores

Período Especial

Aula 4: MEPA

Bruno Müller Junior

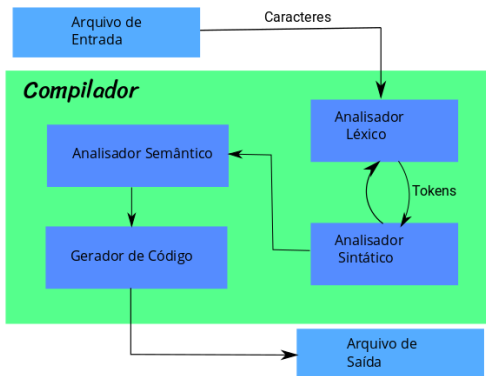
Departamento de Informática
UFPR

2020

- 1 Introdução
- 2 Modelo Esquemático
 - Instruções
 - Constantes e Aritmética
- 3 Roteiro
 - Compilador
- 4 Tradução
 - Regra Sintática (regra 1)
 - Regra Tradução
- 5 Notação
 - Regras que indicam repetição
 - Regras que indicam opcional
 - Tarefas
 - Simulador Mepa
 - Compilador

Compilador

- Partes que Compõe um Compilador



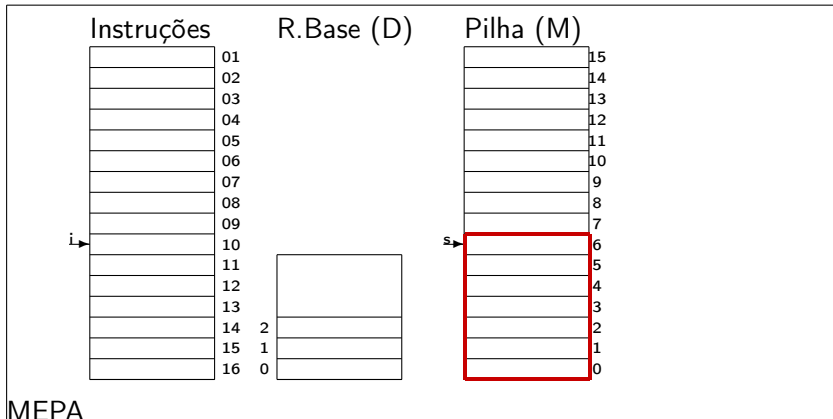
Introdução

- O bison pode gerar código assembly diretamente nos nós “executáveis”, mas não “encaixa” fácil.
- A MEPA (Máquina de Execução de Pascal) é uma linguagem intermediária que simplifica a geração de código.
- Características
 - Máquina que usa uma pilha para cálculos (como na notação posfixa: $abc+-$) e demais operações.
 - Memória (M)
 - Registradores de base (D)
 - Contador de instruções (i)
 - apontador da pilha (s) (*stack pointer*).

Introdução

- Modelo Esquemático da MEPA.
- A pilha cresce para cima.
- Da posição 0 (zero) até o topo (s), a pilha contém valores válidos. De $s + 1$ para cima, são valores desconhecidos ou inválidos.

Modelo Esquemático



Instruções

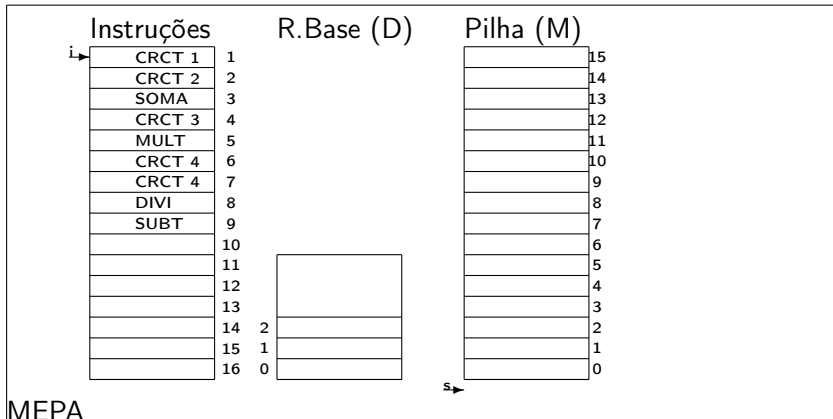
- As instruções da MEPA implementam o modelo de execução.
- Todas elas são descritas com um mnemônico de quatro letras precedidas ou não de um rótulo seguido do símbolo dois pontos (:)
- Estas instruções serão apresentadas gradualmente, mas todas podem ser encontradas no livro do Tomasz (usaremos só o modelo básico).

Constantes e Aritmética

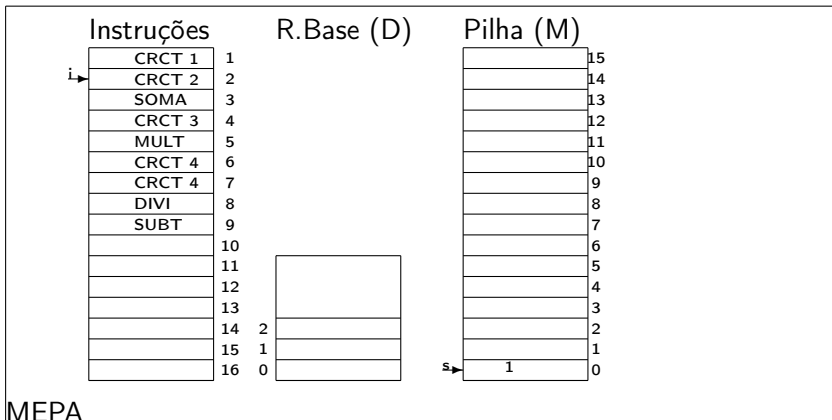
Instrução	Ação	Significado
CRCT k	s:=s+1; M[s]:=k i:=i+1	Carrega Constante
SOMA	M[s-1]:=M[s-1]+M[s]; s:=s-1; i:=i+1	Soma
SUBT	M[s-1]:=M[s-1]-M[s]; s:=s-1; i:=i+1	Subtrai
MULT	...	multiplica
DIVI	...	divide

Exemplo de Tradução	
Expressão	Código MEPA equivalente
(1+2)*3-4/4	CRCT 1
	CRCT 2
	SOMA
	CRCT 3
	MULT
	CRCT 4
	CRCT 4
	DIVI
	SUBT

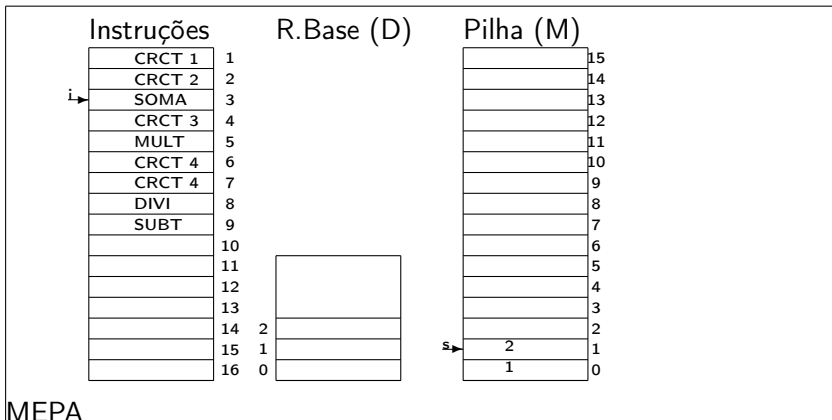
Execução (1)



Execução (2)



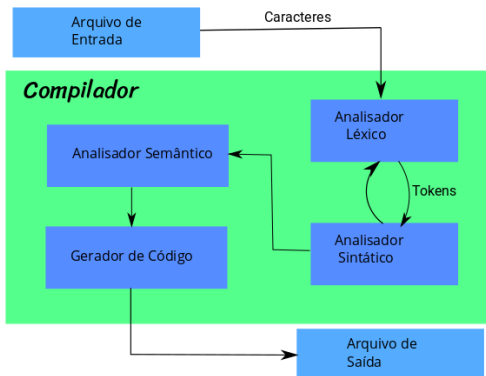
Execução (3)



- Objetivo: Explicar a tradução Pascal \Rightarrow MEPA
- A cada aula, serão explicados os seguintes tópicos:
 - Construção a ser implementada;
 - Esquema de tradução;
 - Regras gramaticais a serem usadas;
 - Adaptação para bison;
- Como cada aula é construída sobre as construções já conhecidas nas aulas anteriores, cada aluno **DEVE** implementar cada aula o quanto antes.
- Regras gramaticais: apêndice B do livro do Tomasz (notação a ser corrigida).

Compilador

- Partes que Compõe um Compilador



Regra Sintática (regra 1)

● Regra 1:

```
program <identificador> (<lista de identificadores>);  
    <bloco>.
```

- O que está escrito em vermelho são tokens;
- O que está entre “<” e “>” são regras;
- Considere que a regra <bloco> está vazia. Assim, uma entrada válida para esta regra é:
program exemplo (input, output); .

- O compilador é um programa que verifica se uma entrada está de acordo com as regras sintáticas e semânticas de uma gramática e se estiver gera código “executável” (que nesta disciplina é o código MEPA).
- As regras sintáticas do bison ajudam nesta geração de código. Um exemplo é quando usar as instruções INPP e PARA.

Instrução	Ação	Significado
INPP	s:=-1; D[0]:=0; i:=i+1	Inicia Programa Pascal
PARA		Finaliza

- Ao encontrar o token `program`, o bison deve imprimir a instrução INPP, e ao encontrar o ponto final, imprimir PARA.

- Regra 1:

program <identificador> (<lista de identificadores>);
 <bloco>.

- Uma sugestão de implementação é:

```
programa :{  
    geraCodigo (NULL, "INPP");  
}  
PROGRAM IDENT  
ABRE_PARENTESES lista_idents FECHA_PARENTESES PONTO_E_VIRGULA  
bloco PONTO  
{  
    finalizaCompilador();  
    geraCodigo (NULL, "PARA");  
}  
;
```

- onde a função geraCodigo(rotulo, comando MEPA) tem dois parâmetros: o rótulo e o comando MEPA a ser gerado no arquivo de saída.

- O apêndice 1 contém todas as regras que usaremos na disciplina (não usaremos as que tem um (*) ao final.
- Porém, há um problema: elas estão num formato não aceito pelo bison, em especial:
- { e } para indicar repetição;
- [e] para indicar opcional;
- A seguir, será explicado como converter as regras que contém estes símbolos nas regras equivalentes, apropriadas ao bison.

- Exemplo de regra que indica repetição

10. `<lista de identificadores> ::= <identificador> {, <identificador>}`

- entrada válida: $\alpha = \text{"a, g1, b"}$.

- Formato livro: $A ::= \beta\{\alpha\}$

- Equivalente em bison: $A ::= A\alpha|\beta$

- Mapeamento:

- $A \Rightarrow \text{< lista de identificadores >}$
- $\alpha \Rightarrow \text{, < identificador >}$
- $\beta \Rightarrow \text{< identificador >}$

- Logo:

10. `<lista de identificadores> ::= <lista de identificadores> VIRGULA <identificador>
| <identificador>`

- Exemplo de regra que indica repetição

10. `<lista de identificadores> ::= <identificador> {, <identificador>}`

- entrada válida: $\alpha = \text{"a, g1, b"}$.

- Formato livro: $A ::= \beta\{\alpha\}$

- Equivalente em bison: $A ::= A\alpha|\beta$

- Mapeamento:

- $A \Rightarrow \text{< lista de identificadores >}$
- $\alpha \Rightarrow \text{, < identificador >}$
- $\beta \Rightarrow \text{< identificador >}$

- Logo:

10. `<lista de identificadores> ::= <lista de identificadores> VIRGULA <identificador>
| <identificador>`

- Exemplo de regra opcional:

2. <bloco> ::= [<parte de declaração de rótulos>] ...

- <parte de declaração de rótulos> é opcional.

- Formato livro: $A ::= [B]$

- Como B é opcional, a regra A pode ser lida como “ou B ou vazio”, ou seja:

```
A           ::= B_ou_vazio
B_ou_vazio ::= B
           | // acrescentando a regra vazia
```

- Isto funciona porque nestes no Pascal simplificado, o primeiro token de B indicará se a regra deve ser usada ou não.

- De <http://www.inf.ufpr.br/bmuller/CI211.html>, baixe o Interpretador MEPA.
- Este arquivo contém o programa que desenvolvi para simular o funcionamento da MEPA.
- Compile (make).
- Verifique o conteúdo de MEPA4.
- Execute o simulador para MEPA4.

- De <http://www.inf.ufpr.br/bmuller/CI211.html>, baixe o arquivo `Projeto.tar.bz2` ;
- Este arquivo contém o início do compilador (arquivos `flex`, `bison`, `header (.h)` e `subrotinas C (.c)`).
- Compile (`make`) e execute o compilador com entradas simples.
- Acrescente a geração de código para `INPP` e `PARA`.

- Página para anotações

Licença

- Slides desenvolvidos somente com software livre:
 - \LaTeX usando beamer;
 - Inkscape.
- Licença:
 - Creative Commons Atribuição-Use Não-Comercial-Vedada a Criação de Obras Derivadas 2.5 Brasil License. <http://creativecommons.org/licenses/by-nc-nd/2.5/br/>