Construção de Compiladores
Período Especial
Aula 5: Variáveis & MEPA

Bruno Müller Junior

Departamento de Informática
UFPR

2020

Objetivos

- O escopo das variáveis da linguagem Pascal seguem um modelo bem definido, porém nem sempre intuitivo.
- Por esta razão, antes de explicar como implementar a declaração e uso de variáveis, iremos:
  - relembrar o escopo das variáveis em Pascal;
  - apresentar como isto é mapeado na MEPA;
  - traduzir & executar um programa Pascal simples na MEPA;
  - apresentar a tabela de símbolos.

Escopo das Variáveis

- O programa abaixo tem duas variáveis globais.

```
1 program declaraVars (input, output);
2     var a, b:  integer
3     begin
4       a:=a+b;
5     end.
```

- Uma forma de acessá-las seria utilizar instruções que referenciam diretamente o endereço de cada variável na memória.
- Primeiras linguagens faziam isto;
- Problema: variáveis de procedimentos e funções recursivas.

Exemplo

```
1   program variaveis (input, output);
2   var a, b:  integer
3         procedure p(x,y:integer);
4         var a:integer
5         begin
6               ...
7         end
8         procedure q(a:integer);
9               procedure r(a:integer);
10              begin
11              ...
12              end
13        begin
14              ...
15        end
16  begin
17        a:=a+b;
18  end.
```

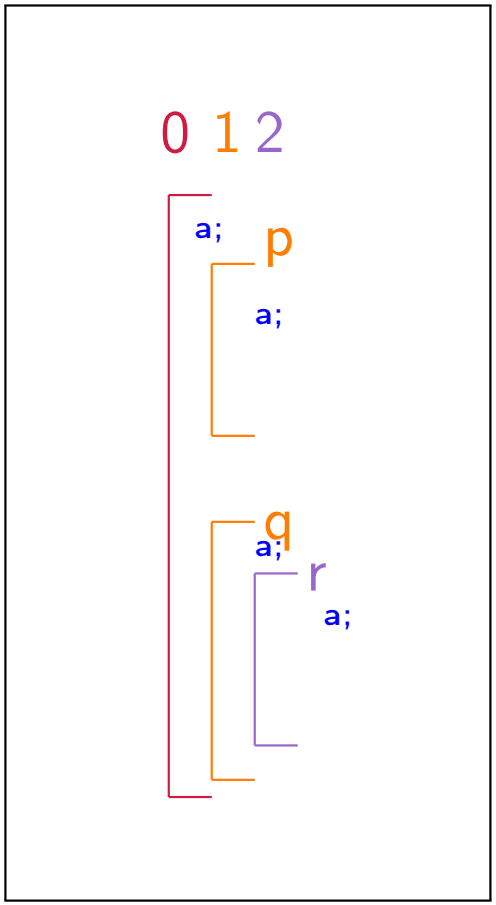
- "a" e "b" são visíveis em todos os lugares, porém nem sempre o mesmo "a".
- Não podem ser mapeados no mesmo local.
- *Até que podem, mas é complicado*.
- A solução baseia-se no escopo da variável.

Exemplo de Programa

```
1  program variaveis (input, output);
2  var a, b:  integer
3       procedure p(x,y:integer);
4       var a:integer
5       begin
6            ...
7       end
8       procedure q(a:integer);
9            procedure r(a:integer);
10           begin
11           ...
12           end
13      begin
14           ...
15      end
16 begin
17      a:=a+b;
18 end.
```

- Neste programa, as chaves indicam o escopo das variáveis.
- As "colunas" indicam o nível léxico:
  - Principal: 0
  - p e q: 1
  - r: 2

Esquema



- Várias instâncias de “a”.
- Instâncias $\times$ nível léxico.
- Idéia: colocar o nível léxico como parte do endereço de cada variável.
- aloca espaço na entrada do procedimento;
- libera espaço na saída.

MEPA

- O acesso às variável na MEPA utiliza dois parâmetros: nível léxico e deslocamento.
- O espaço para as variáveis é criado na pilha: basta acrescentar espaço usando o registrador s da MEPA.

Instrução	Ação	Significado
AMEM m	`s:=s+m;`	Aloca Memória
DMEM m	`s:=s-m;`	Desaloca Memória
CRVL k,n	`s:=s+1;` `M[s]:=M[D[k]+n];`	Carrega Valor
ARMZ k,n	`M[D[k]+n]:=M[s];` `s:=s-1`	Armazena Valor

Esquema

- A idéia é usar o Vetor de Registradores de Base (D) como apontador do registro de ativação corrente daquele nível.
- Assim, `D[0]` aponta para o R.A. ativo de nível 0, `D[1]` para o R.A. de nível 1, ... , `D[k]` para o R.A. de nível k.
- Por esta razão, nas instruções do tipo `CRVL k,n` os parâmetros `k,n` são conhecidos como "endereço léxico" da variável.

Exemplo de Tradução

```
program varsGlobais (input, output);
var a, b: integer
begin
   a:=0;
   b:=a+10;
end.
```

Exemplo de Tradução

```
program varsGlobais (input, output);                INPP
var a, b: integer
begin
   a:=0;
   b:=a+10;
end.
```

Exemplo de Tradução

```
program varsGlobais (input, output);
var a, b: integer;
begin
   a:=0;
   b:=a+10;
end.
```

```
INPP
AMEM 2
```

Exemplo de Tradução

```
program varsGlobais (input, output);
var a, b: integer;
begin
   a:=0;
   b:=a+10;
end.
```

```
INPP
AMEM 2
CRCT 0
ARMZ 0,0
```

Exemplo de Tradução

```
program varsGlobais (input, output);
var a, b: integer;
begin
   a:=0;
   b:=a+10;
end.
```

```
INPP
AMEM 2
CRCT 0
ARMZ 0,0
CRVL 0,0
CRCT 10
SOMA
ARMZ 0,1
```

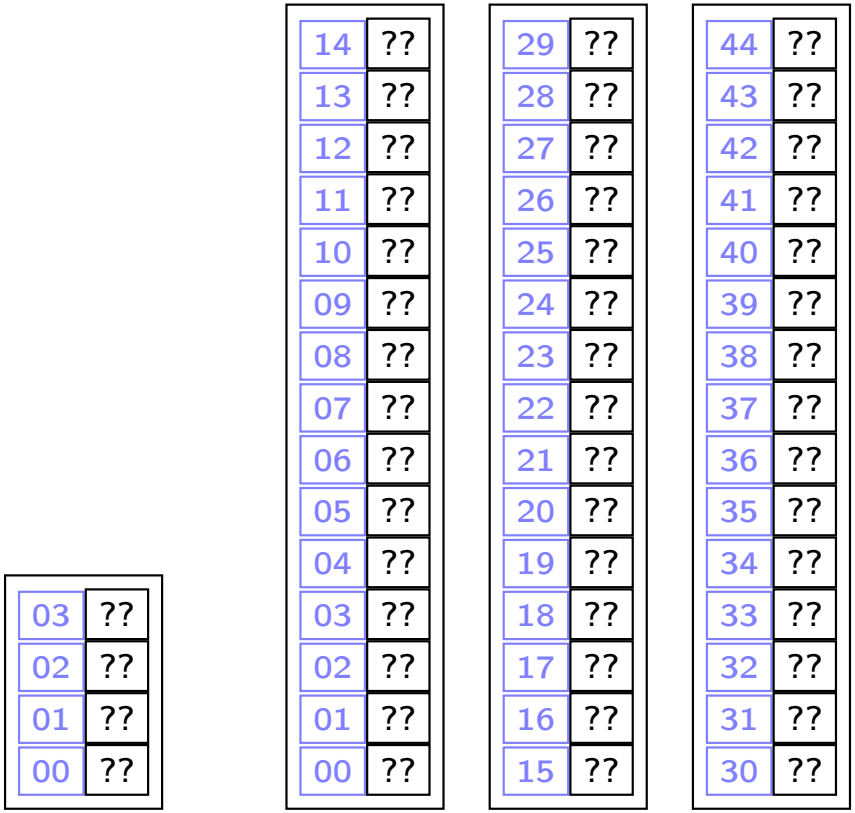
Exemplo de Tradução

```
program varsGlobais (input, output);
var a, b: integer;
begin
   a:=0;
   b:=a+10;
end.
```

```
INPP
AMEM 2
CRCT 0
ARMZ 0,0
CRVL 0,0
CRCT 10
SOMA
ARMZ 0,1
DMEM 2
PARA
```

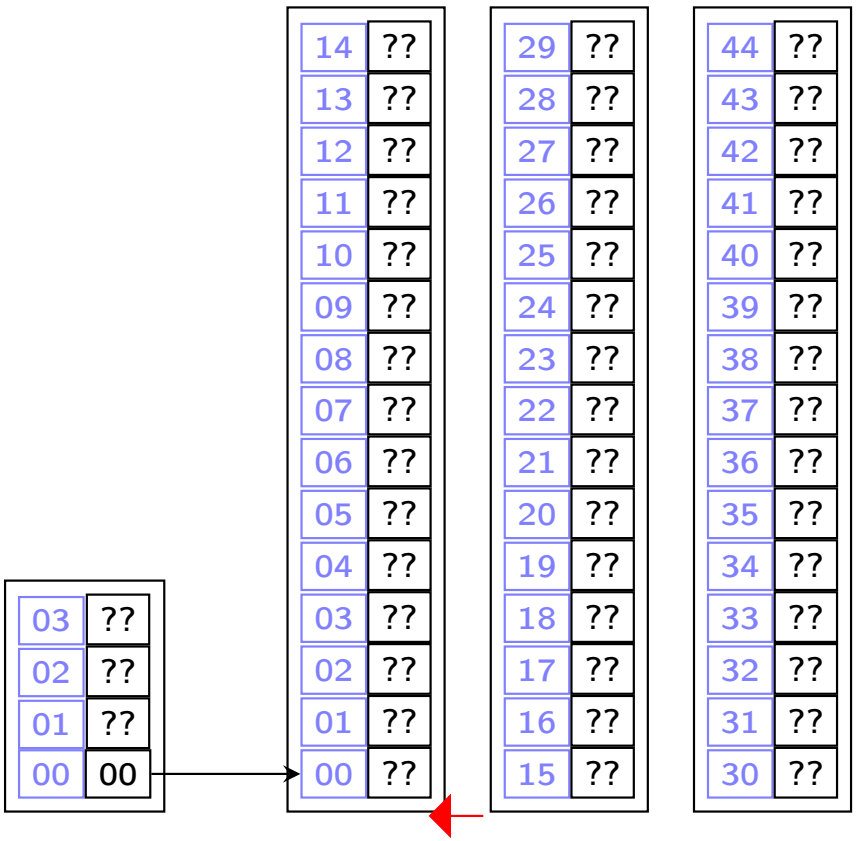
Simulação da Execução na MEPA

```
INPP
AMEM 2
CRCT 0
ARMZ 0,0
CRVL 0,0
CRCT 10
SOMA
ARMZ 0,1
DMEM 2
PARA
```



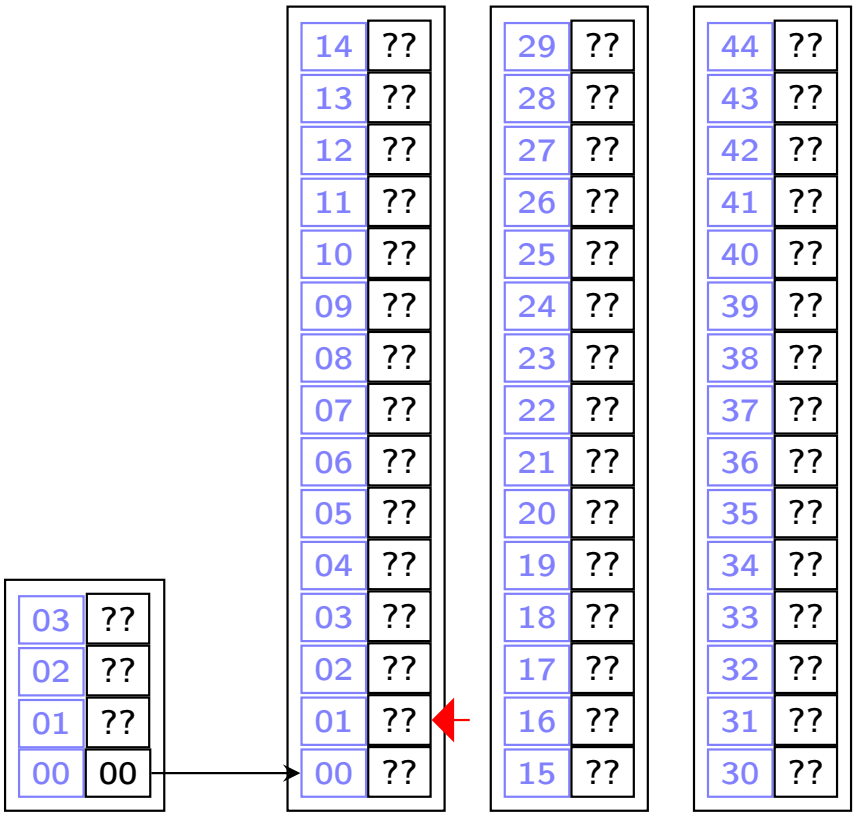
Simulação da Execução na MEPA

```
INPP        D[0]:=0; s:=-1
AMEM 2
CRCT 0
ARMZ 0,0
CRVL 0,0
CRCT 10
SOMA
ARMZ 0,1
DMEM 2
PARA
```



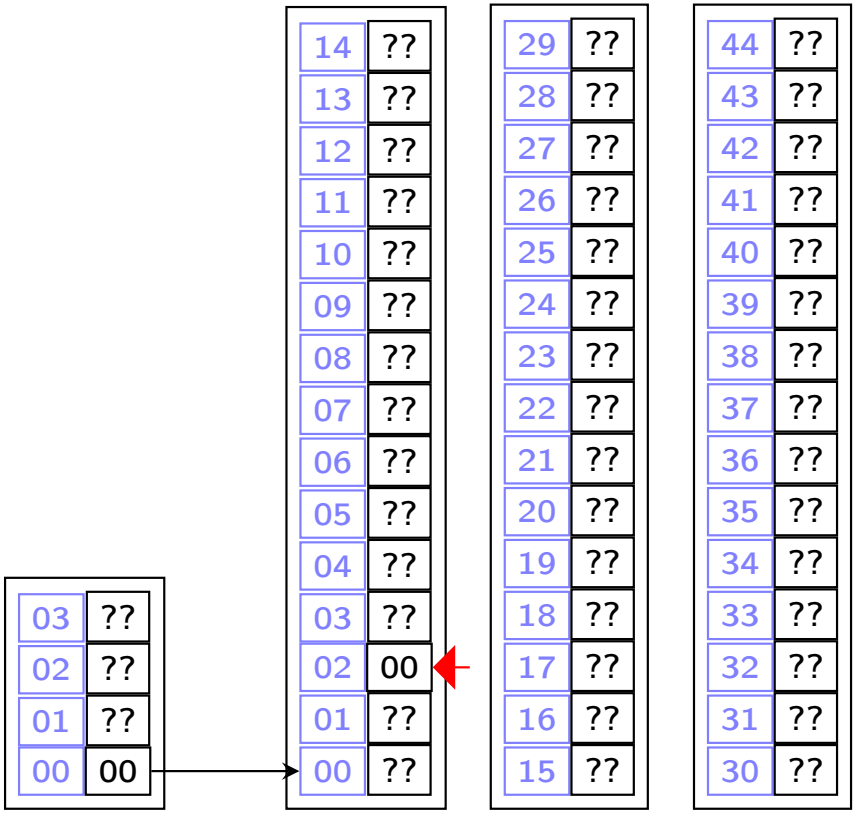
Simulação da Execução na MEPA

```
INPP
AMEM 2      s:=s+2
CRCT 0
ARMZ 0,0
CRVL 0,0
CRCT 10
SOMA
ARMZ 0,1
DMEM 2
PARA
```



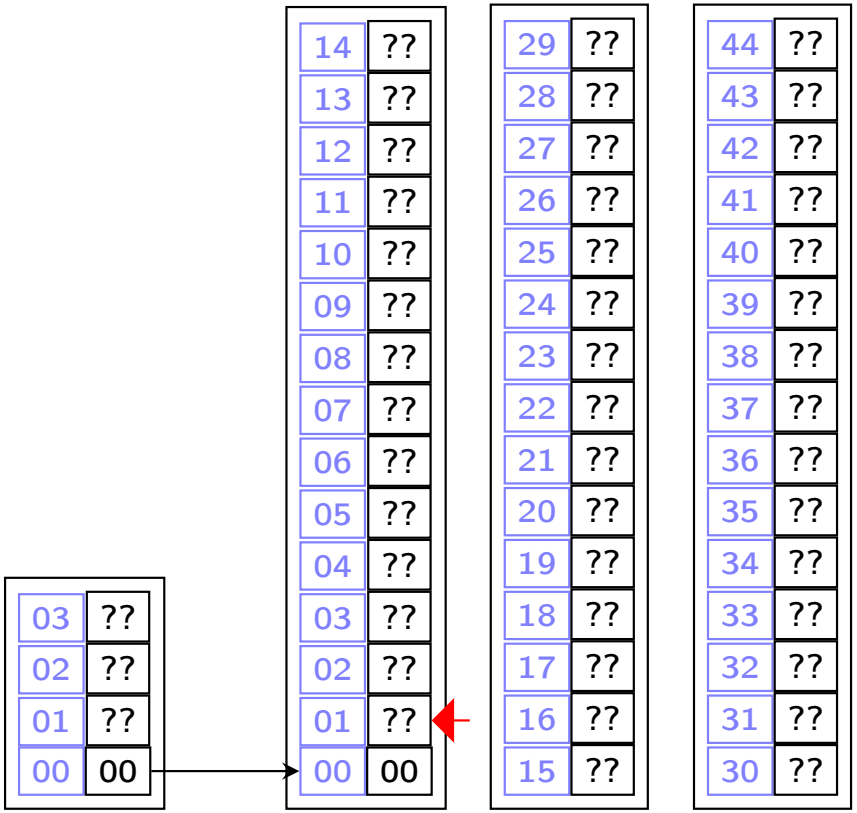
Simulação da Execução na MEPA

```
INPP
AMEM 2
CRCT 0      s:=s+1; M[s]:=0;
ARMZ 0,0
CRVL 0,0
CRCT 10
SOMA
ARMZ 0,1
DMEM 2
PARA
```



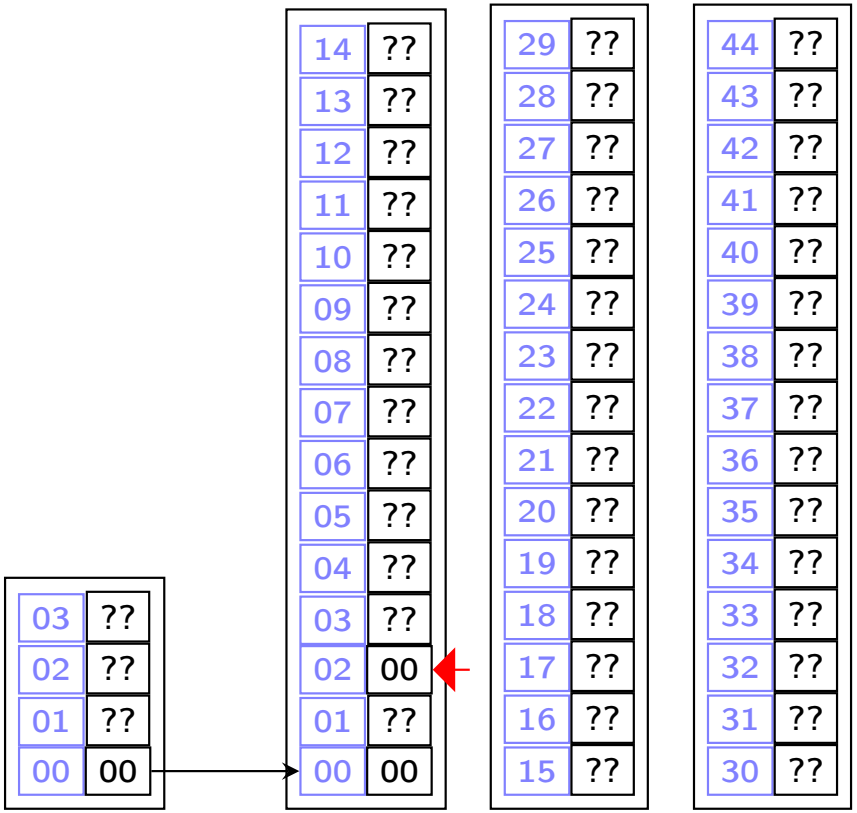
Simulação da Execução na MEPA

```
INPP
AMEM 2
CRCT 0
ARMZ 0,0  M[D[0]+0] := M[s]; s:=s-1
CRVL 0,0
CRCT 10
SOMA
ARMZ 0,1
DMEM 2
PARA
```



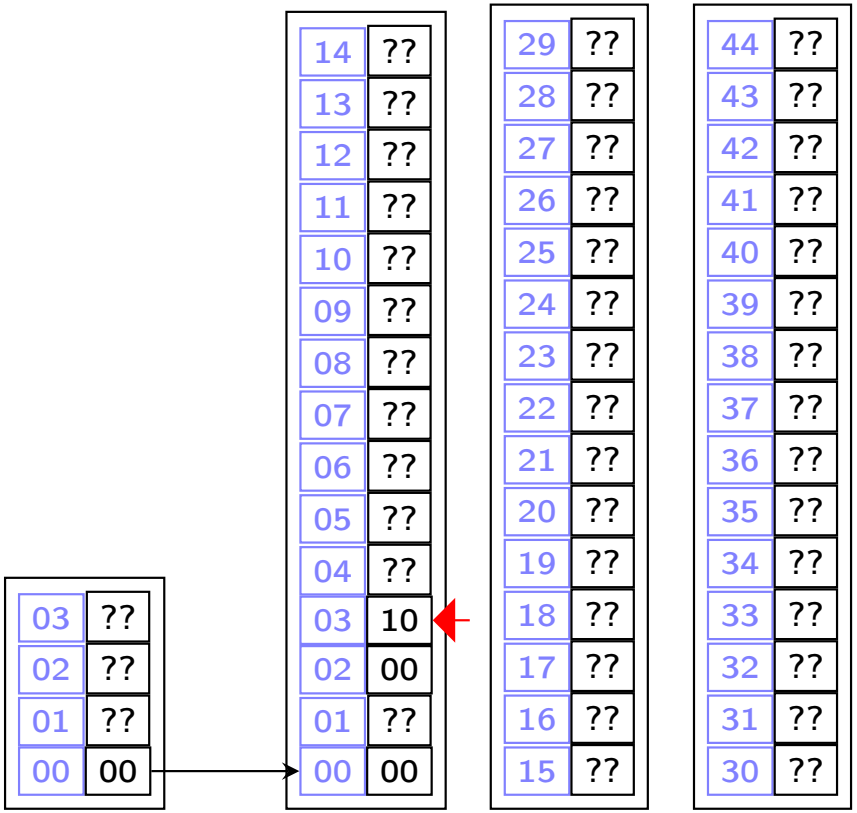
Simulação da Execução na MEPA

```
INPP
AMEM 2
CRCT 0
ARMZ 0,0
CRVL 0,0   s:-s+1; M[s]:=M[D[0]+0]
CRCT 10
SOMA
ARMZ 0,1
DMEM 2
PARA
```



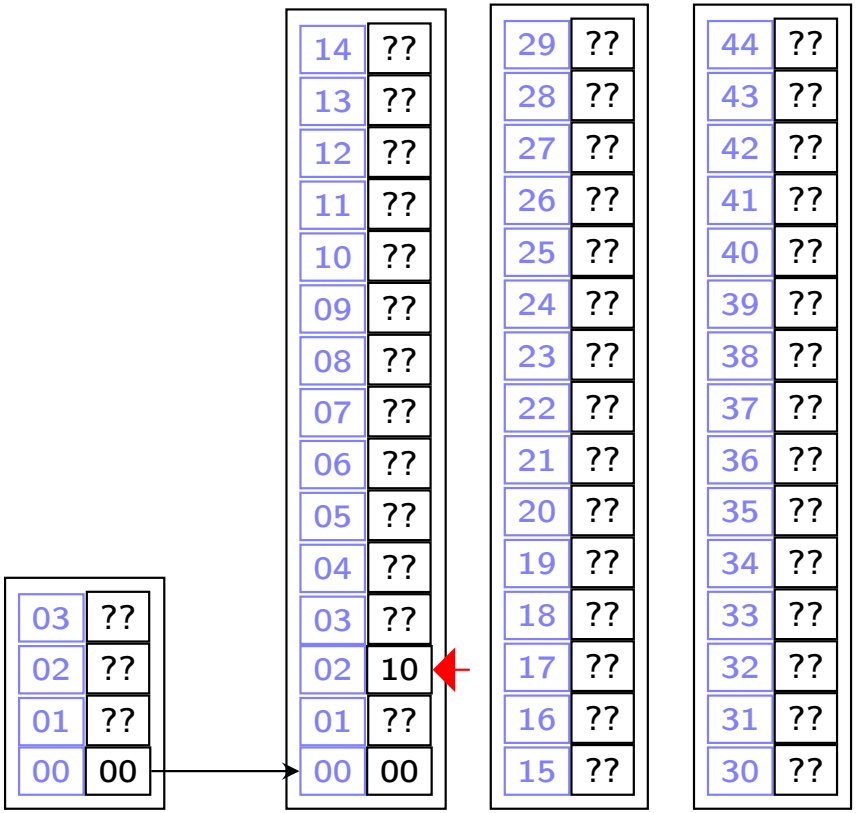
Simulação da Execução na MEPA

```
INPP
AMEM 2
CRCT 0
ARMZ 0,0
CRVL 0,0
CRCT 10    s:=s+1; M[s]:=10
SOMA
ARMZ 0,1
DMEM 2
PARA
```



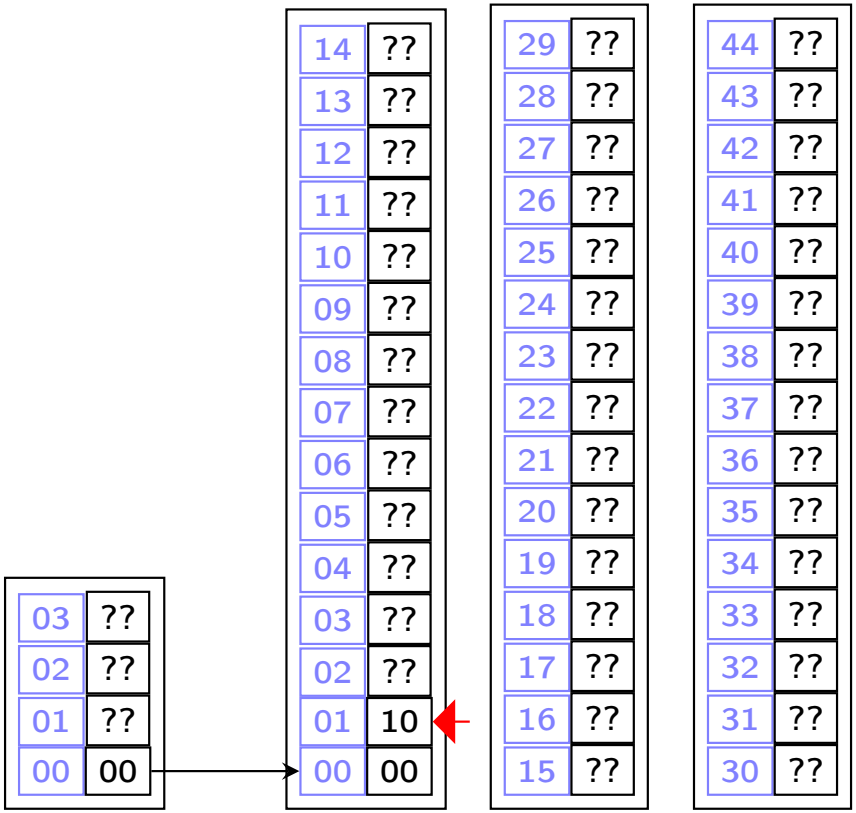
Simulação da Execução na MEPA

```
INPP
AMEM 2
CRCT 0
ARMZ 0,0
CRVL 0,0
CRCT 10
SOMA        M[s-1]:=M[s-1]+M[s]. s:=s-1;
ARMZ 0,1
DMEM 2
PARA
```



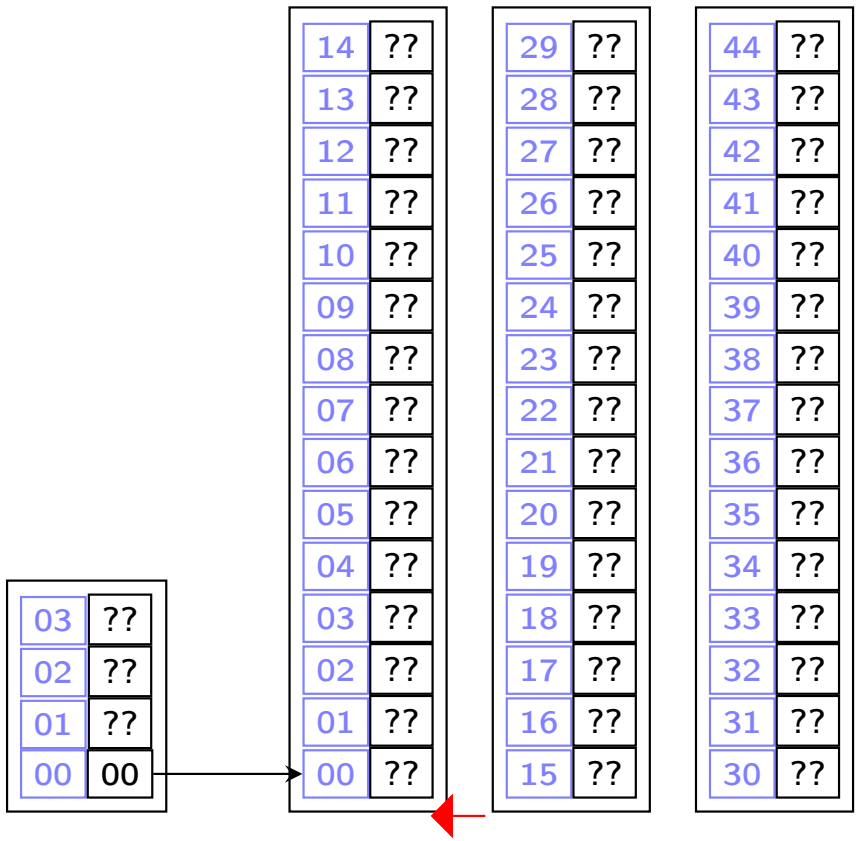
Simulação da Execução na MEPA

```
INPP
AMEM 2
CRCT 0
ARMZ 0,0
CRVL 0,0
CRCT 10
SOMA
ARMZ 0,1   M[D[0]+1]:=M[s]; s:=s-1;
DMEM 2
PARA
```



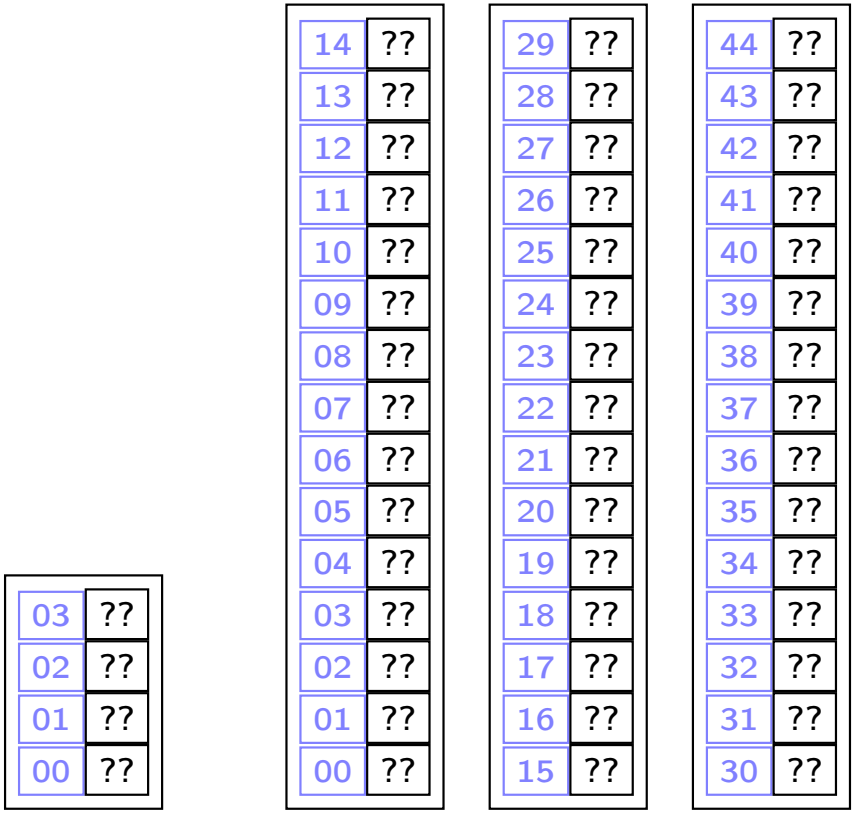
Simulação da Execução na MEPA

```
INPP
AMEM 2
CRCT 0
ARMZ 0,0
CRVL 0,0
CRCT 10
SOMA
ARMZ 0,1
DMEM 2    s:=s-2;
PARA
```



Simulação da Execução na MEPA

```
INPP
AMEM 2
CRCT 0
ARMZ 0,0
CRVL 0,0
CRCT 10
SOMA
ARMZ 0,1
DMEM 2
PARA
```



Compilador

```
program varsGlobais (input, output);
var a, b: integer
begin
   a:=0;
   b:=a+10;
end.
```

- Problema: como saber o endereço léxico das variáveis?
- Declaradas em `var` e usadas entre `begin` e `end`.
- Solução: tabela de símbolos;
- insere na tabela no `var`;
- o que inserir?
- retira da tabela no `end`;

Tabela de Símbolos

- Todos os tokens reconhecidos como identificadores são chamados *símbolos*.
- Símbolos válidos são aqueles que já foram declarados (por exemplo na declaração "var");
- Quando entra em um procedimento, os símbolos lá declarados são alocados, e quando sai devem ser desalocados.
- Confira novamente no programa.
- Observe que a alocação e a desalocação de símbolos seguem a estrutura de uma pilha.

Tabela de Símbolos

- A tabela de símbolos é uma estrutura de dados de um compilador que armazena os símbolos "válidos" para uso naquele momento.
- Se um identificador não está presente na tabela de símbolos, então isto significa que não foi declarado.
- As operações básicas são:

Tabela de Símbolos

- A tabela de símbolos é uma estrutura de dados de um compilador que armazena os símbolos "válidos" para uso naquele momento.
- Se um identificador não está presente na tabela de símbolos, então isto significa que não foi declarado.
- As operações básicas são:

Tabela de Símbolos

- A tabela de símbolos é uma estrutura de dados de um compilador que armazena os símbolos "válidos" para uso naquele momento.
- Se um identificador não está presente na tabela de símbolos, então isto significa que não foi declarado.
- As operações básicas são:

Tabela de Símbolos

- A tabela de símbolos é uma estrutura de dados de um compilador que armazena os símbolos "válidos" para uso naquele momento.
- Se um identificador não está presente na tabela de símbolos, então isto significa que não foi declarado.
- As operações básicas são:

  Insere (ident, atributos) Insere o identificador indicado na TS, assim como seus atributos;

Tabela de Símbolos

- A tabela de símbolos é uma estrutura de dados de um compilador que armazena os símbolos "válidos" para uso naquele momento.
- Se um identificador não está presente na tabela de símbolos, então isto significa que não foi declarado.
- As operações básicas são:

  Insere (ident, atributos) Insere o identificador indicado na TS, assim como seus atributos;

  Busca(ident) Retorna a entrada (os atributos) da TS associados ao ident procurado.

Tabela de Símbolos

- A tabela de símbolos é uma estrutura de dados de um compilador que armazena os símbolos "válidos" para uso naquele momento.
- Se um identificador não está presente na tabela de símbolos, então isto significa que não foi declarado.
- As operações básicas são:

  Insere (ident, atributos) Insere o identificador indicado na TS, assim como seus atributos;

  Busca(ident) Retorna a entrada (os atributos) da TS associados ao ident procurado.

  Retira(n) Retira as últimas *n* entradas da TS.

Categorias

- Os atributos que a T.S. deve armazenar dependem do símbolo, ou melhor, da categoria a que o símbolo pertence.
- O livro do Tomasz (cap.10) descreve as categorias de símbolos que serão utilizadas.
- Os atributos são aqueles que são utilizados, por exemplo na geração de código.

Categorias

- Os atributos que a T.S. deve armazenar dependem do símbolo, ou melhor, da categoria a que o símbolo pertence.
- O livro do Tomasz (cap.10) descreve as categorias de símbolos que serão utilizadas.
- Os atributos são aqueles que são utilizados, por exemplo na geração de código.

Categorias

- Os atributos que a T.S. deve armazenar dependem do símbolo, ou melhor, da categoria a que o símbolo pertence.
- O livro do Tomasz (cap.10) descreve as categorias de símbolos que serão utilizadas.
- Os atributos são aqueles que são utilizados, por exemplo na geração de código.

Categorias

- Os atributos que a T.S. deve armazenar dependem do símbolo, ou melhor, da categoria a que o símbolo pertence.
- O livro do Tomasz (cap.10) descreve as categorias de símbolos que serão utilizadas.
- Os atributos são aqueles que são utilizados, por exemplo na geração de código.

  Variável Simples {nível léxico, tipo, deslocamento}

Categorias

- Os atributos que a T.S. deve armazenar dependem do símbolo, ou melhor, da categoria a que o símbolo pertence.
- O livro do Tomasz (cap.10) descreve as categorias de símbolos que serão utilizadas.
- Os atributos são aqueles que são utilizados, por exemplo na geração de código.

  Variável Simples {nível léxico, tipo, deslocamento}
  Procedimento {...}
  Função {...}
  Parâmetro Formal {...}
  Rótulo {...}

Exemplo de inserção de símbolos na T.S.

```
1   program variaveis (input, output);
2   var a, b:  integer
3       procedure p(x,y:integer);
4       var a:integer
5       begin
6           ...
7       end
8       procedure q(a:integer);
9           procedure r(a:integer);
10          begin
11          ... r; ... p; ...
12          end
13      begin
14          ... r ...
15      end
16  begin
17      ... q ...
18  end.
```

Compilador

- Consulte o capítulo 10 do livro do Tomasz, em especial no que ele referencia a TS. Atenção às sugestões de implementação.
- Implemente um Tipo Abstrato de Dados "Tabela de Símbolos" (TS).
- Ela deve funcionar como uma pilha na inserção e na remoção.
- A busca deve procurar do **último** para o `primeiro`.
- A remoção simplesmente altera o topo da pilha.

- Página para anotações

Licença

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