

# Um compilador simples de uma passagem

Máquinas de pilha abstratas

**Prof. Edson Alves**

Faculdade UnB Gama

## Máquinas de Pilha Abstratas

- ▶ A interface de vanguarda do compilador produz uma representação intermediária do programa fonte, que será usada pela interface de retaguarda para produzir o programa alvo
- ▶ Uma possível forma para a representação intermediária é a máquina de pilha abstrata
- ▶ Uma máquina de pilha abstrata possui memórias separadas para dados e instruções, e todas as operações aritméticas são realizadas sobre os valores em uma pilha
- ▶ As instruções são divididas em três classes: aritmética inteira, manipulação de pilha e fluxo de controle
- ▶ O ponteiro *pc* indica qual é a próxima instrução a ser executada

## Instruções aritméticas

- ▶ A máquina de pilha abstrata precisa implementar cada operador da linguagem intermediária
- ▶ Operações elementares, como adição e subtração, são suportadas diretamente
- ▶ Operações mais sofisticadas devem ser implementadas como uma sequência de instruções da máquina
- ▶ A título de simplificação, assuma que existe uma instrução para cada operação aritmética
- ▶ O código de uma máquina de pilha abstrata para uma expressão simula a avaliação de uma representação posfixa, usando uma pilha
- ▶ A avaliação segue da esquerda para a direita, empilhando os operandos
- ▶ Quando um operador é encontrado, seus operandos são extraídos da pilha (do último para o primeiro), a operação é realizada e o resultado é inserido no topo da pilha

# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

**Instrução**

**Ações**

**Pilha**

\_\_\_\_\_

# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

Instrução

Ações

1 2 + 3 \*

↑

Pilha

\_\_\_\_\_

# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

Instrução

Ações

1 2 + 3 \*  
↑

Empilhar o valor 1

Pilha

\_\_\_\_\_

# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

Instrução

Ações

1 2 + 3 \*  
↑

Empilhar o valor 1

Pilha



# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

**Instrução**

**Ações**

1 2 + 3 \*

↑

**Pilha**





# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

**Instrução**

1 2 + 3 \*  
↑

**Ações**

Empilhar o valor 2

**Pilha**



# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

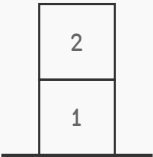
Instrução

Ações

1 2 + 3 \*  
↑

Empilhar o valor 2

Pilha



# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

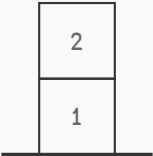
Instrução

Ações

1 2 + 3 \*

↑

Pilha



# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

Instrução

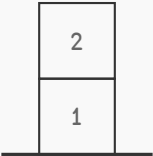
Ações

1 2 + 3 \*



Adicionar 1+2

Pilha



# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

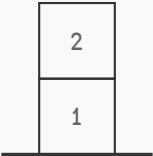
**Instrução**

1 2 + 3 \*  
↑

**Ações**

Adicionar 1+2  
Empilhar a soma

**Pilha**



# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

Instrução

1 2 + 3 \*  
↑

Ações

Adicionar 1+2  
Empilhar a soma

Pilha



# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

**Instrução**

1 2 + 3 \*

↑

**Ações**

Empilhar o valor 3

**Pilha**



# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

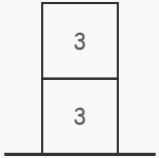
**Instrução**

1 2 + 3 \*  
          ↑

**Ações**

Empilhar o valor 3

**Pilha**





# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

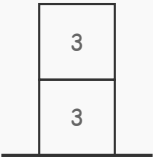
**Instrução**

**Ações**

1 2 + 3 \*



**Pilha**



# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

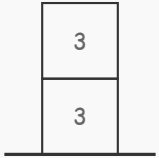
**Instrução**

1 2 + 3 \*  
          ↑

**Ações**

Multiplicar 3\*3

**Pilha**



# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

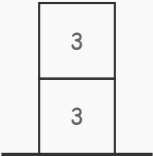
Instrução

1 2 + 3 \*  
          ↑

Ações

Multiplicar  $3*3$   
Empilhar o produto

Pilha



# Exemplo de avaliação da expressão $12+3*$ por máquina abstrata de pilha

**Instrução**

1 2 + 3 \*  
          ↑

**Ações**

Multiplicar  $3*3$   
Empilhar o produto

**Pilha**



## Valores-L e valores-R

- ▶ O significado de um identificador depende da posição onde ele ocorre em uma atribuição
- ▶ No lado esquerdo, o identificador se refere à localização de memória onde o valor deve ser armazenado
- ▶ No lado direito, o identificador se refere ao valor armazenado na localização de memória associada ao identificador
- ▶ Valor-L e valor-R se referem aos valores apropriados para os lados esquerdo e direito de uma atribuição, respectivamente
- ▶ Um mesmo identificador pode ser um valor-L e um valor-R na mesma atribuição (por exemplo, o identificador  $x$  em  $x = x + 1$ )

## Manipulação da pilha

Uma máquina de pilha abstrata suporta as seguintes instruções para a manipulação da pilha:

Instrução	Significado
<code>push <math>v</math></code>	empilha $v$
<code>pop</code>	desempilha o valor do topo da pilha
<code>valor-r <math>p</math></code>	empilha o valor armazenado no endereço de memória $p$
<code>valor-l <math>p</math></code>	empilha o endereço de memória $p$
<code>:=</code>	o valor-R do topo da pilha é armazenado no valor-L do subtopo (elemento que está abaixo do topo) da pilha
<code>copiar</code>	empilha o valor do topo da pilha

## Tradução de expressões

- ▶ O código para avaliar uma expressão na máquina de pilha abstrata tem uma relação direta com a notação posfixa
- ▶ Por exemplo, a expressão  $a + b$  é traduzida para o código intermediário

```
valor-r  a
valor-r  b
+
```

- ▶ As atribuições são traduzidas da seguinte maneira: o valor-L do identificador é empilhado, a expressão à direita é avaliada e o seu valor  $r$  é atribuído ao identificador
- ▶ Formalmente,

$$cmd \rightarrow \mathbf{id} := expr \{ cmd.t := \text{"valor-l"} \parallel \mathbf{id}.lexema \parallel expr.t \parallel \text{" := "} \}$$

## Tradução da expressão $F = 9 * C / 5 + 32$ para máquina de pilha abstrata

```
1  valor-l    F
2  push      9
3  valor-r    C
4  *
5  push      5
6  /
7  push      32
8  +
9  :=
```



## Fluxo de controle

- ▶ A máquina de pilha abstrata executa as instruções sequencialmente, na ordem em que foram dadas
- ▶ Há três formas de se especificar um desvio (salto) no fluxo de execução
  1. o operando da instrução fornece o endereço da localização alvo;
  2. o operando da instrução fornece a distância relativa (positiva ou negativa) a ser saltada
  3. o alvo é especificado simbolicamente (a máquina deve suportar rótulos)
- ▶ Nas duas primeiras formas, uma alternativa é especificar que o salto está no topo da pilha
- ▶ A terceira forma é a mais simples de se implementar, pois não há necessidade de se recalculer os endereços caso o número de instruções seja modificado durante a otimização

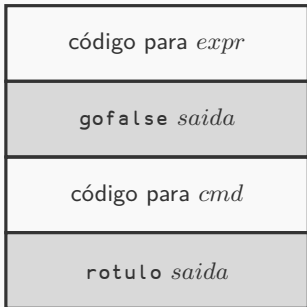
## Instruções de fluxo de controle

Uma máquina de pilha abstrata suporta as seguintes instruções para o fluxo de controle:

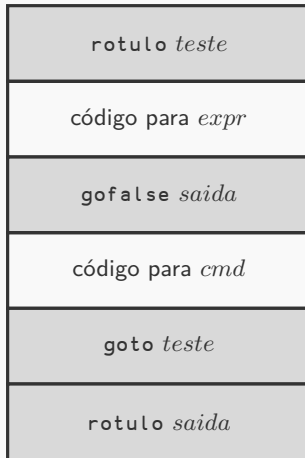
Instrução	Significado
<code>rótulo <math>r</math></code>	especifica o rótulo $r$ como possível alvo de desvios; não há outros efeitos
<code>goto <math>r</math></code>	a próxima instrução é tomada a partir do rótulo $r$
<code>gofalse <math>r</math></code>	desempilha o topo da pilha: salta para $r$ se o valor for igual a zero
<code>gotrue <math>r</math></code>	desempilha o topo da pilha: salta para $r$ se o valor for diferente de zero
<code>parar</code>	encerra a execução

## Gabaritos para tradução de condicionais e laços

**if**



**while**



## Unicidade dos rótulos

- ▶ Os rótulos *saida* e *teste* que ilustram os gabaritos das condicionais e dos laços devem ser únicos, para evitar possíveis ambiguidades
- ▶ É preciso, portanto, de uma estratégia que torne tais rótulos únicos durante a tradução
- ▶ Seja *novoRotulo* um procedimento que gera, a cada chamada, um novo rótulo único
- ▶ A ação semântica associada ao comando **if** assumiria a seguinte forma:

$$\begin{aligned} cmd \rightarrow \text{if } expr \text{ then } cmd_1 \{ & \textit{saida} := \textit{novoRotulo}; \\ & \textit{cmd.t} := \textit{expr.t} \\ & || \text{"gofalse"} \textit{saida} \\ & || \textit{cmd}_1.t \\ & || \text{"rotulo"} \textit{saida} \} \end{aligned}$$


## Referências

1. **AHO**, Alfred V, **SETHI**, Ravi, **ULLMAN**, Jeffrey D. *Compiladores: Princípios, Técnicas e Ferramentas*, LTC Editora, 1995.
2. GNU.org. [GNU Bison](#), acesso em 23/05/2022.
3. Wikipédia. [Flex \(lexical analyser generator\)](#), acesso em 23/05/2022.