

Lógica da Computação - 2019/2

Roteiro 8 - Simple Calculator **v3.0**

Raul Ikeda - rauligs@insper.edu.br

Entrega: 2019/2

Nome:

Objetivos

1. Geração de Código

Exemplo de Código:

```
Sub Main()  
  Dim i as Integer  
  Dim n as Integer  
  Dim f as Integer  
  n = 5  
  i = 2  
  f = 1  
  While i < n + 1  
    f = f * i  
    i = i + 1  
  Wend  
  Print f  
End Sub
```

Mesmo exemplo em Assembly NASM (gerado pelo compilador):

```
; constantes  
SYS_EXIT equ 1  
SYS_READ equ 3  
SYS_WRITE equ 4  
STDIN equ 0  
STDOUT equ 1  
True equ 1  
False equ 0  
  
segment .data  
  
segment .bss ; variaveis  
  res RESB 1  
  
section .text  
  global __start  
  
print: ; subrotina print  
  
  PUSH EBP ; guarda o base pointer
```

```

MOV EBP, ESP ; estabelece um novo base pointer

MOV EAX, [EBP+8] ; 1 argumento antes do RET e EBP
XOR ESI, ESI

print_dec: ; empilha todos os digitos
MOV EDX, 0
MOV EBX, 0x000A
DIV EBX
ADD EDX, '0'
PUSH EDX
INC ESI ; contador de digitos
CMP EAX, 0
JZ print_next ; quando acabar pula
JMP print_dec

print_next:
CMP ESI, 0
JZ print_exit ; quando acabar de imprimir
DEC ESI

MOV EAX, SYS_WRITE
MOV EBX, STDOUT

POP ECX
MOV [res], ECX
MOV ECX, res

MOV EDX, 1
INT 0x80
JMP print_next

print_exit:
POP EBP
RET

; subrotinas if/while
binop_je:
JE binop_true
JMP binop_false

binop_jg:
JG binop_true
JMP binop_false

binop_jl:
JL binop_true
JMP binop_false

binop_false:
MOV EBX, False
JMP binop_exit

binop_true:
MOV EBX, True

binop_exit:
RET

__start:

PUSH EBP ; guarda o base pointer

```

```

MOV EBP, ESP ; estabelece um novo base pointer

; codigo gerado pelo compilador

PUSH DWORD 0 ; Dim i as Integer [EBP-4]
PUSH DWORD 0 ; Dim n as Integer [EBP-8]
PUSH DWORD 0 ; Dim f as Integer [EBP-12]

MOV EBX, 5
MOV [EBP-8], EBX ; n = 5

MOV EBX, 2
MOV [EBP-4], EBX ; i = 2

MOV EBX, 1
MOV [EBP-12], EBX ; f = 1

LOOP_34:
MOV EBX, [EBP-4]
PUSH EBX ; empilha i

MOV EBX, [EBP-8]
PUSH EBX ; empilha n
MOV EBX, 1
POP EAX
ADD EAX, EBX ; n + 1
MOV EBX, EAX

POP EAX
CMP EAX, EBX
CALL binop_jl ; i < n + 1
CMP EBX, False
JE EXIT_34

MOV EBX, [EBP-12]
PUSH EBX ; empilha f
MOV EBX, [EBP-4]
POP EAX ; empilha i
IMUL EBX ; i * f
MOV EBX, EAX
MOV [EBP-12], EBX ; f = f * i
MOV EBX, [EBP-4]
PUSH EBX ; empilha i
MOV EBX, 1
POP EAX
ADD EAX, EBX ; i + 1
MOV EBX, EAX
MOV [EBP-4], EBX ; i = i + 1
JMP LOOP_34
EXIT_34:

MOV EBX, [EBP-12]
PUSH EBX ; empilha f
CALL print ; Print f
POP EBX ; limpa args

; interrupcao de saida
POP EBP
MOV EAX, 1
INT 0x80

```

Técnicas

- Para os nós que dependem de filhos que retornam valor (*UnOp*, *BinOp*, etc), assumir que o *Evaluate* do filho esquerdo irá para o *EAX* via pilha e o do filho direito irá para o *EBX*. Para pais que possuem mais de 2 filhos, utilizar a pilha.
- Alguns comandos (*if*, *while*, etc) dependem de um *label* para marcar pontos de retorno ou avanço. Criar um identificador único para cada nó da AST e ao criar o *label*, utilizar o identificador para gerar um *label* único.
- Agora a *SymbolTable* precisa guardar quantos bytes a variável estará deslocada de EBP. Assumir *Integer* como *DWORD* (4 bytes) e *Boolean* também como *DWORD*. Note que esse valor é acumulativo.

Tarefas

As tarefas vão se concentrar praticamente no analisador semântico:

- Criar um atributo estático *i* na classe *Node*. Criar um método estático *newId()* que vai incrementar o atributo estático *i* e retornar o novo valor.
- Alterar o construtor para criar um atributo com o retorno da função *newId()*. Se a linguagem que você utiliza não roda o construtor da classe base automaticamente, você deve fazê-lo manualmente nas classes filhas.
- Modificar o *Evaluate()* para gerar código para as seguintes operações:
 - Declaração de variáveis
 - Operações aritméticas
 - Atribuição
 - Condicional
 - Loop
 - Print
- Você pode aproveitar as subrotinas do código de apoio do exemplo acima (print e condicionais) publicadas no Blackboard
- Para montar o código e transformá-lo em executável no Linux/x86:

```
$ nasm -f elf32 -F dwarf -g program.asm
$ ld -m elf_i386 -o program program.o
```

Para mais informações: https://en.wikipedia.org/wiki/Executable_and_Linkable_Format

Em caso de problemas com o assembly, use o **gdb**.

Questionário

1. Você deve ter percebido que o código é bem estruturado mas pouco eficiente. Proponha otimizações na geração de código.

2. Proponha modificações na linguagem para incorporar funções e escopo de variáveis.