Geração de código para L1

Construção de compiladores I

Objetivos

Objetivos

- Apresentar a implementação de um gerador de código de máquina virtual para L1

Motivação

Motivação

- Na aula anterior, vimos como implementar um verificador de tipos para a linguagem L3.
- Nesta aula, vamos mostrar como implementar a geração de código para L1.

Geração de código para L1

Geração de código para L1

- Bastante similar a geração de código para expressões aritméticas em L0.
- O que muda?
  - Geração de código para atribuição.
  - Geração de código para read / print.

Geração de código para L1

- Gerando código para atribuições

```
s1Codegen (LAssign v e1)
  = e1Codegen e1 ++ [Store v]
```

Geração de código para L1

- Gerando código para read / print

```
s1Codegen (LRead s v)
  = [Push (VStr s), Print, Input, Store v]
s1Codegen (LPrint e1)
  = e1Codegen e1 ++ [Print]
```

Geração de código para L1

- Geração de código C

```
cL1Codegen :: L1 -> String
cL1Codegen e
  = unlines $ [ "#include <stdio.h>"
              , "// code generated for expressions"
              , "int main () {" ] ++
              (map (nest 3) (generateBody e)) ++
              [ nest 3 "return 0;"
              , "}"
              ]
    where
      nest n v = replicate n ' ' ++ v
```

Geração de código para L1

- Geração de código C

```
generateStmt :: S1 -> String
generateStmt (LAssign v e1)
  = unwords ["int", pretty v, "=", generateExp e1, ";"]
generateStmt (LPrint e1)
  = unwords ["printf(%d,", generateExp e1, ");"]
generateStmt (LRead s v)
  = unwords ["print(\"",s,"\");\n", "scanf(%d, &", pretty v, ")"]
```

Conclusão

Conclusão

- Apresentamos a geração de código para a linguagem simples, L1.
  - Máquinas virtual com memória e pilha.
  - Geração de código C.