Compiladores

prof. Ricardo Oliveira

2023/2 - **Trabalho da disciplina**

Enunciado

Invente uma nova linguagem de programação, e implemente um **compilador** que compila programas da sua linguagem para o *assembly* Pilheitor (descrito na próxima seção).

Com a linguagem de programação que você inventar, deve ser possível:

- declarar variáveis inteiras;
- ler variáveis (do usuário) e imprimir seus valores (para o usuário);
- escrever expressões aritméticas (com soma, subtração, multiplicação, divisão inteira, resto da divisão e parênteses); imprimi-las e atribuí-las a variáveis;
- escrever seleções (equivalente a "if"/"else") e laços (equivalente a "while"), inclusive aninhados, cujas condições envolvam os operadores de comparação: maior, menor, maior-ou-igual, menor-ou-igual, igual e diferente;
- escrever comentários de linha (o equivalente a "//").

A sua linguagem não precisa fornecer: variáveis não inteiras (float/char/etc); estruturas de dados (vetor/matriz/struct); ponteiros/endereçamento de memória; funções; operadores lógicos (and/or/not); controle de laços (break/continue); demais elementos não listados acima.

Alguns exemplos de programas que leem duas variáveis e imprime a maior delas, escritas em linguagens inventadas, são:

```
programa maior:
decls [
    var1 <- 0 ;     @ declarando vars
    evair <- 0 ;
]
ler var1 ;          @ lendo as vars e pa
ler evair ;
se var1 > evair fazer {
    printar var1 ;
} senaum {
    printar evair ;
}
```

```
inteiro var1 comecacom 0 .      %% declarando vars
inteiro evair comecacom 0 .
medah var1 .
medah evair .                    %% lendo as vars e pa
testase ( var1 > evair ) sefor
comeca
    memostra var1 .
termina
senaofor
comeca
    memostra evair .
termina
fimtesta
```

Veja uma outra linguagem inventada que ficou famosa: https://birl-language.github.io/ .

Seja criativo!

Pilheitor

O Pilheitor é uma máquina simples, cuja memória consiste nos registradores %0, %1, %2, %3, ... e em uma (única) **pilha**. Seu *assembly* consiste nas seguintes instruções:

Instrução	Descrição	Obs
`PUSH v` ou `PUSH %r`	Empilha o valor v ou o conteúdo do registrador %r	push(v) ou push(%r)
`ATR %r`	Desempilha o topo da pilha e o atribui ao registrador %r	%r ← pop()
`LEIA`	Lê um inteiro do usuário e o empilha	push(input())
`IMPR`	Desempilha o topo da pilha e o imprime ao usuário	print(pop())
`SOMA`	Desempilha b, desempilha a e empilha $a + b$	b=pop();a=pop();push(a+b)
`SUB`	Desempilha b, desempilha a e empilha $a - b$	b=pop();a=pop();push(a-b)
`MULT`	Desempilha b, desempilha a e empilha $a \times b$	b=pop();a=pop();push(a*b)
`DIV`	Desempilha b, desempilha a e empilha a/b (divisão inteira)	b=pop();a=pop();push(a/b)
`MOD`	Desempilha b, desempilha a e empilha $a\%b$ (resto da divisão)	b=pop();a=pop();push(a%b)
`MAIOR`	Desempilha b, desempilha a e empilha 1 se $a > b$ ou 0 caso contrário	b=pop();a=pop();push(a>b)
`MENOR`	Desempilha b, desempilha a e empilha 1 se $a < b$ ou 0 caso contrário	b=pop();a=pop();push(a<b)
`MAIOREQ`	Desempilha b, desempilha a e empilha 1 se $a \geq b$ ou 0 caso contrário	b=pop();a=pop();push(a≥b)
`MENOREQ`	Desempilha b, desempilha a e empilha 1 se $a \leq b$ ou 0 caso contrário	b=pop();a=pop();push(a≤b)
`IGUAL`	Desempilha b, desempilha a e empilha 1 se $a = b$ ou 0 caso contrário	b=pop();a=pop();push(a==b)
`DIFER`	Desempilha b, desempilha a e empilha 1 se $a \neq b$ ou 0 caso contrário	b=pop();a=pop();push(a≠b)
`GFALSE rot`	Desempilha c e, se $c = 0$, vai para o rótulo *rot*	"Goto on false"
`GTRUE rot`	Desempilha c e, se $c \neq 0$, vai para o rótulo *rot*	"Goto on true"
`GOTO rot`	Vai para o rótulo *rot* (não altera a pilha)	Lit. "Goto"
`NADA`	Não faz nada	feijoada
`SAIR`	Termina a execução	Obrigatório ser a última instrução

Um rótulo é descrito por `rot:` antes de uma instrução.

Como exemplo, considere os seguintes programas em *Assembly* do Pilheitor. O programa à esquerda lê do usuário dois números nos registradores %0 e %1, e imprime o valor de (%0+%1)/2. Já o programa da direita inicializa os registradores; lê seus valores do usuário; determina e imprime o maior deles (os programas dados de exemplo na primeira seção, compilados no *Assembly* do Pilheitor, resultam neste programa).

```
LEIA
ATR %0
LEIA
ATR %1
PUSH %0
PUSH %1
SOMA
PUSH 2
DIV
IMPR
SAIR
```

```
     PUSH 0
     ATR %0
     PUSH 0
     ATR %1
     LEIA
     ATR %0
     LEIA
     ATR %1
     PUSH %0
     PUSH %1
     MAIOR
     GFALSE R00
     PUSH %0
     IMPR
     GOTO R01
R00: NADA
     PUSH %1
     IMPR
R01: NADA
     SAIR
```

Ainda como exemplo, o programa à esquerda imprime todos os inteiros de 1 a 100. O programa à direita lê uma sequência de inteiros (terminada em -1) e, para cada inteiro lido, imprime sua metade (se par) ou seu dobro (se impar).

```
     PUSH 1
     ATR %0
R00: NADA
     PUSH %0
     PUSH 100
     MENOREQ
     GFALSE R01
     PUSH %0
     IMPR
     PUSH %0
     PUSH 1
     SOMA
     ATR %0
     GOTO R00
R01: NADA
     SAIR
```

```
     LEIA
     ATR %0
R00: NADA
     PUSH %0
     PUSH -1
     DIFER
     GFALSE R01
     PUSH %0
     PUSH 2
     MOD
     PUSH 0
     IGUAL
     GFALSE R02
     PUSH %0
     PUSH 2
     DIV
     IMPR
     GOTO R03
R02: NADA
     PUSH %0
     PUSH 2
     MULT
     IMPR
R03: NADA
     LEIA
     ATR %0
     GOTO R00
R01: NADA
     SAIR
```

Você pode executar os programas em *Assembly* do Pilheitor com o interpretador de Pilheitor que está em anexo ao trabalho (compilar `pilheitor.cpp` e executar `./pilheitor programa.pil`).

Implementação

- Seu compilador pode escrito em C, C++ ou Python*;
- Você pode usar qualquer técnica de *parser* que preferir na sua implementação;
- Você pode implementar o compilador manualmente, ou usar geradores de analisadores como as ferramentas flex/bison (para C/C++) ou a biblioteca PLY* (para Python);
- Você pode restringir alguns elementos da sua linguagem inventada para facilitar a implementação (por exemplo: *obrigar* que variáveis declaradas sejam inicializadas; que blocos tenham delimitadores (como `{` e `}` em C/C++) mesmo que só tenham uma única linha de código; etc);
- Seu compilador deve apontar a ocorrência de erros (léxicos, sintáticos ou semânticos) em programas que não compilam. Basta indicar o tipo de erro ocorrido, não sendo necessário descrever o erro em si (isto é, basta indicar "erro léxico", "erro sintático" ou "erro semântico").

Orientações

- O trabalho pode ser feito por equipes de *no máximo* 3 (três) estudantes;
- Submeta, via *Moodle*, um pacote **zip** ou **tar.gz** contendo todo o código-fonte necessário para compilar e executar sua solução, além de um arquivo de texto (txt) onde conste:
  - O nome de todos os integrantes da equipe;
  - A gramática, em BNF, da sua linguagem;
  - Toda informação que a equipe julgar relevante para a correção (como *bugs* conhecidos, detalhes de implementação, escolhas de projeto, etc.)
- Comente adequadamente seus códigos para facilitar a correção;
- O trabalho deve ser entregue até **11 de Dezembro de 2023, 23:59**, apenas via *Moodle*. Trabalhos entregues por outros meios ou fora do prazo não serão aceitos. É suficiente que o trabalho seja submetido por apenas um estudante da equipe;
- Trabalhos detectados como cópia, plágio (de colegas ou da internet) ou comprados receberão **todos** a nota 0 (**ZERO**) e estarão sujeitos a abertura de Processo Administrativo Disciplinar Discente.