ESCOLA DE CIÊNCIAS E TECNOLOGIA
DEPARTAMENTO DE INFORMÁTICA

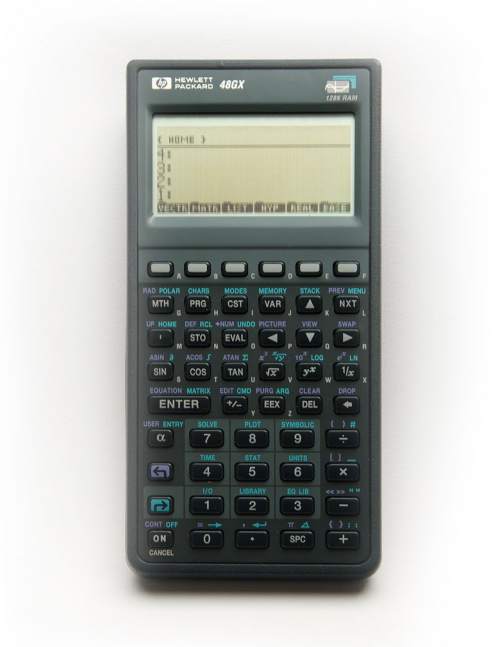
Trabalho de Arquitectura de Sistemas e Computadores I

Licenciatura em Engenharia Informática

2017–2018

1 Objectivo

Pretende-se desenvolver uma calculadora que opere na notação polonesa inversa (*RPN - Reverse Polish Notation*). A calculadora deverá receber uma ou mais strings da consola, realizar as operações lá indicadas e mostrar na consola o estado da memória da calculadora na forma de uma pilha.



2 Notação polonesa inversa

A notação polonesa inversa, também conhecida como notação pós-fixada, é uma notação onde os operandos são introduzidos primeiro, seguidos dos operadores. Enquanto na notação infixa o operador é colocado no meio dos operandos, como em $3 + 4$, na notação pósfixa o operador surge no final, como em 3 4 +.

Uma calculadora em notação polonesa funciona com uma pilha. Os operandos, representados a verde, são colocados na pilha e os operadores, representados a azul, são aplicados sobre o topo da pilha, substituindo os valores usados pelo resultado da operação.

Para realizar a operação $(3 + 4) \times 5$ na notação polonesa inversa escreve-se 3 4 + 5 ×. Note que a precedência marcada pelos parentesis é satisfeita. Caso se pretendesse calcular $3 + (4 \times 5)$, na notação polonesa inversa escrever-se-ia 3 4 5 × +.

Para além dos operadores binários, que recebem dois argumentos, há também necessidade de operadores unários que apenas têm um operando. Um exemplo é o cálculo do simétrico de um número, habitualmente escrito como -3. Em notação polonesa inversa escreve-se como `3 neg`, sendo `neg` uma operação distinta da subtracção[1].

Um operador unário é aplicado ao topo da pilha, enquanto um binário é aplicado aos dois operandos do topo da pilha.

A tabela seguinte mostra alguns exemplos de expressões nas notações infixa e pósfixa:

Notação infixa	Notação pósfixa	Comentário
$3-4$	`3 4 -`	subtracção
-3	`3 neg`	o operador unário `neg` calcula o simétrico
$-3+4$	`3 neg 4 +`	
$(3+4)\times 5$	`3 4 + 5 ×`	A soma é efectuada primeiro
$3+4\div 2$	`3 4 2 ÷ +`	A divisão é efectuada primeiro
3×3	`3 dup ×`	A operador `dup` duplica (clone) o topo da pilha
sqrt(3)	`3 sqrt`	Calcula raiz quadrada
sqrt(3) $\div$ 3	`3 dup sqrt swap ÷`	O operador `swap` troca os operandos do topo da pilha

Para exemplificar os vários passos intermédios da execução, considere-se a expressão $1+\frac{\sqrt{3}}{3}$ que em notação polonesa inversa se escreve

`1 3 dup sqrt swap ÷ +`

A tabela seguinte ilustra a execução desta expressão à medida que os dados vão sendo introduzidos. A coluna com o estado da pilha mostra os dados separados por vírgulas, sendo o lado direito o topo da pilha.

Dados inseridos	Estado da pilha
	(vazia)
`1`	1
`3`	1, 3
`dup`	1, 3, 3
`sqrt`	1, 3, 1.732
`swap`	1, 1.732, 3
`÷`	1, 0.577
`+`	1.577

A notação polonesa tem várias vantagens como o facto de não requerer parêntesis e das operações poderem ser directamente executadas à medida que vão sendo lidas, o que não acontece na notação infixa.

As calculadores RPN foram popularizadas pela Hewlett-Packard nos anos 60 e continuam a existir nos dias de hoje vários modelos em comercialização nas versões de calculadoras genéricas, científicas e financeiras. Mais informação em http://en.wikipedia.org/wiki/Reverse_Polish_notation.

3 Especificação do problema

O trabalho a desenvolver consiste num programa em assembly MIPS para correr no simulador MARS.

O programa deve receber strings da consola contendo expressões em notação polonesa inversa, efectuar os cálculos respectivos mantendo o estado numa pilha, e mostrando a pilha na consola.

Os dados podem ser inseridos separados por espaços ou quebras-de-linha. Cada vez que é dada uma quebra-de-linha (tecla *enter*) deve ser mostrado o estado da pilha na consola. No caso dos operandos e operadores serem separados por espaços o resultado só é mostrado no final da execução da expressão.

A pilha deve ser apresentada na vertical sendo os últimos números (mais de baixo) o topo da pilha. Em seguida mostra-se um exemplo de uma sessão com o *input* do utilizador assinalado a vermelho.

[1] Os operadores escritos como sequências letras são escritos apenas neste enunciado com sombreado para facilitar a leitura.

```
*** Calculadora em notação polonesa inversa          ***
*** Autores: Joao (123), Maria (456) e Manuel (789) ***
*** ASC1 2017/2018                                   ***

Pilha:
(vazia)

-> 2
Pilha:
2

-> 3
Pilha:
2
3

-> swap
Pilha:
3
2

-> -
Pilha:
1

-> clear
Pilha:
(vazia)
```

No exemplo anterior, os dados foram inseridos um a um, podendo observar-se todos os estados intermédios da pilha. Alternativamente, o utilizador deverá poder escrever numa única linha todos os operandos e operadores separados por um espaço. Neste caso apenas é apresentado o resultado final.

```
*** Calculadora em notação polonesa inversa          ***
*** Autores: Joao (123), Maria (456) e Manuel (789) ***
*** ASC1 2017/2018                                   ***

Pilha:
(vazia)

->  2 3 swap -
Pilha:
1

-> clear
Pilha:
(vazia)
```

Obviamente, o utilizador pode em qualquer momento escrever linhas só com um operando/operador, ou com vários.

A calculadora deverá funcionar sobre números inteiros e deverá suportar, no mínimo, os seguintes operadores:

Operador	Descrição
`+`	Adição (operador binário)
`-`	Subtracção (operador binário)
`*`	Multiplicação (operador binário)
`/`	Divisão (operador binário)
`neg`	Calcula o simétrico (operador unário)
`swap`	Troca posição dos dois operandos do topo da pilha
`del`	Remove um operando do topo da pilha (em caso de engano)
`clear`	Limpa toda a pilha (fica vazia)
`dup` ou linha vazia	Duplica (clone) o operando do topo da pilha
`off`	Desliga a calculadora (termina o programa)

O suporte de vírgula flutuante é opcional e fica ao critério dos alunos decidir se é ou não implementado.

4 Trabalho a desenvolver

O programa deverá estar devidamente estruturado em funções e seguir as convenções MIPS para o uso dos registos, chamada de funções, passagem de argumentos, pilha, variáveis locais, *etc.*

O trabalho deverá ser realizado em grupos de dois alunos e ser entregue em três fases:

Arquitectura geral do programa Nesta fase deverá apresentar um documento escrito com a organização do programa em funções e mostrando o *loop* principal do programa. Não deve conter código assembly, mas pode incluir, se assim o entender, pseudo-código, ou código numa linguagem de alto nível e fluxogramas. Deve descrever as estruturas de dados usadas e a sua operação.

Versão *beta* Nesta fase deverá apresentar um programa com as principais funções implementadas.

Versão final Esta é a versão final do trabalho, que deverá ser acompanhado de um relatório em formato PDF, preferencialmente escrito em LaTeX, com a descrição detalhada da organização do código e a listagem de funções implementadas. As funções devem estar documentadas de forma semelhante às *man pages* acessíveis no terminal em Linux/Unix.

A entrega das várias versões é feita no `http://moodle.uevora.pt` nas datas indicadas.

Bom trabalho!
Miguel Barão