UNIVERSIDADE DE ÉVORA

RPN - Reverse Polish Notation
Calculadora em Assembly MIPS

André Rato nº45517
José Alexandre nº45223

Maio
Ano Letivo 2019/2020

Arquitetura de Sistemas e Computadores I

Prof. Miguel Barão

Índice

1 Introdução

Este relatório consiste na explicação da organização do nosso trabalho. À semelhança com a calculadora feita na linguagem C, este programa terá que ler *strings* e analisá-las caractere a caractere, de modo a operar segundo as instruções do utilizador.

2 Registos Utilizados

Registos	Descrição
`$a0`	Guarda os endereços a serem usados como argumentos na função `syscall`
`$a1`	Guarda os valores a serem usados como argumentos na função `syscall`
`$s0`	Registo auxiliar de comparação usado na função print
`$s1`	Registo auxiliar usado para percorrer a pilha na função print
`$s2`	Guarda o endereço da pilha quando esta tem só um elemento
`$s3`	Guarda o endereço da pilha quando esta tem dois elementos
`$s4`	Guarda o endereço da pilha quando esta tem zero elementos
`$s5`	Guarda o endereço da pilha quando esta tem dez elementos
`$s6`	Guarda o endereço inicial da input
`$s7`	Registo que guarda o endereço da pilha onde se colocará um novo número
`$t0`	Registo auxilixar usado na conversão de números em *string* para inteiros
`$t1`	Registo auxiliar usado na colocação de elementos na pilha (se `$t1=1` coloca)
`$t2`	Guarda o byte que está a ser analisado
`$t3`	Registo auxiliar para retirar elementos da pilha nas diversas operações
`$t4`	Registo auxiliar para retirar elementos da pilha nas diversas operações
`$t5`	Registo auxiliar usado na conversão dos números (`$t3=10`)
`$v0`	Guarda os valores a serem usados como discriminação da função `syscall`

3 Função Principal - Main

A *main* é a função principal. É nela que são escolhidos alguns registos (`$s2, $s3, $s4, $s5` e `$s7`) que serão utilizados para controlo do número de elementos (`$s2, $s3, $s4` e `s5`) e para controlo da pilha em si (`$s7`).
Depois desta inicialização, é mostrado ao utilizador o cabeçalho (label `intro`), seguida da stack, inicialmente vazia. Após isto, o programa entra num `ciclo`.

4 Ciclo de Leitura da Input

É neste ciclo (`loop`) que é lida a *input* do utilizador, e guardada no espaço alocado para ela (`input: .space 64`), através do comando `syscall` com o valor 8 em `v0`. Antes de o programa entrar num novo ciclo (`loop`), o valor presente em `a0` é movido para o registo `$v0`, de modo a guardar o endereço da `input` para ser utilizado mais tarde, e os endereços `$t0` e `$t1` são inicializados com 0, valores estes que terão grande importância quando a função `numeros` for chamada.

5 Ciclo de Análise da Input

Aqui é feita toda a análise da `input`. Para a execção desta análise, a `input` é lida caractere a caractere, à semelhança do que foi feito na calculadora em C. Se o caractere lido corresponder a algum dos caracteres atribuídos a funções (`+, -, *, /`), o programa realizará a função correspondente. Caso encontre um caractere, cujo código ASCII esteja compreendido entre 48 e 57 (inclusíve), o programa irá fazer a conversão dos mesmo para inteiro, utilizando a seguinte fórmula: `t0 = t0 * 10 + (t2 - 48)`. Deste modo, é permitida a colocação deste números na pilha.
Se o caractere lido for uma letra, o programa analisará uma sequência de letras e comparará a outras sequências predefinidas associadas às operações (`neg` para a negação, `swap` para a troca, `clear` para a limpeza da pilha, `dup` para clonar, `drop` para remover o elemento da pilha, `off` para desligar a calculadora e `help` para mostrar todos os comandos). Caso o caractere lido seja um espaço ou uma quebra de linha ('\n'), o programa verifica se o caractere anterior é um número ou não, e se for, coloca-o na `stack`.
Se não houver correspondência com nenhum deste fatores, é apresentada a mensagem "Unkown command. Type 'help' for available commands", voltando ao início do `ciclo`.

6 Funções

6.1 Funções soma, subt, prod e divi

Caso o caractere presente em `$t1` (caractere a ser analisado) seja `'+', '-', '*'` ou `'/'`, o programa realiza a função correspondente.
Começa por verificar se o número de elementos na pilha é suficiente para realizar a operação (`blt $s7,$s3, elementosInsuficientes`), e caso sejam insuficientes, apresenta a mensagem "*Not enought values in the stack!*".
Caso os elementos sejam suficientes, o programa retira os dois últimos elementos da pilha e coloca o resultado na mesma.
No caso da divisão (`divi`), se o último elemento for 0 (divisor = 0), o programa mostra a mensagem "*Error! Division by 0 is impossible!*".

6.2 Função nega

Caso o caractere presente em `$t1` seja `'n'`, o programa verifica se os próximos caracteres na input são `'e'` e `'g'`.
Caso isso aconteça verifica ainda se o caractere seguinte é `'\n'` ou `' '`. Se isso não acontecer o programa mostra a mensagem *"Unkown command. Type 'help' for available commands!"*.
Se acontecer o programa retira o valor do topo da *stack*, caso exista, e coloca o simétrico desse valor de volta na pilha.

6.3 Função swap

Caso o caractere presente em `$t1` seja `'s'`, o programa verifica se os próximos caracteres na input são `'w', 'a'` e `'p'`.
Caso isso aconteça verifica ainda se o caractere seguinte é `'\n'` ou `' '`. Se isso não

acontecer o programa mostra a mensagem *"Unkown command. Type 'help' for available commands!"*.
Se acontecer o programa retira os dois valores do topo da *stack*, caso existam, e coloca-os de volta na pilha pela ordem contrária.

6.4 Função clear

Caso o caractere presente em `$t1` seja `'c'`, o programa verifica se os próximos caracteres na input são `'l', 'e', 'a'` e `'r'`.
Caso isso aconteça verifica ainda se o caractere seguinte é `'\n'`. Se isso não acontecer o programa mostra a mensagem *"Unkown command. Type 'help' for available commands!"*.
Se acontecer o programa coloca $s7 no valor inicial (endereço da label `pilha`), resetando assim a pilha. Se a pilha já estiver vazia, o programa mostra a mensagem "*Stack is already empty!*".

6.5 Função "funcoesd"

Nesta função, são avaliadas duas possibilidades em relação a operações: podemos ter a função `dup` e a função `drop`. Desde modo, os caracteres são avaliados, à semelhança do resto das funções (caractere a caractere, e confirmando se após ambas as palavras o caractere seguinte é '\n' ou ' ').
Caso a expressão lida seja "dup", o programa recua uma posição na pilha, retira para $t3 o último valor da pilha, avança para a posição em que estava e volta a colocar o mesmo valor na pilha, avançando, novamente, uma posição na pilha.
Já no caso de a expressão ser "drop", o programa retrocede apenas uma posição na pilha, dando parecendo assim que o valor foi removido.
Ambas estas duas funções avaliam se existem elementos suficientes para operar. Caso não existam, o programa apresenta a mensagem "*Not enough values in the stack*".

6.6 Função help

Caso o caractere lido seja `'h'` e os seguintes `'e', 'l'` e `'p'`, seguidos de `'\n'`, o programa mostra uma lista de comandos (*string* guardada na label `hlptxt`). Depois disto, o programa volta para o `ciclo`, pronto para receber outra input do utilizador.

6.7 Função espaco

Se o caractere lido for um espaço (' '), o programa verifica duas condições: verifica se o caractere anterior é um número ou não, e verifica se a pilha está cheia ou não (se já contém 10 elementos).
Caso o caractere anterior seja um número e a pilha não esteja cheia, o programa coloca o número na pilha e dá reset às duas "variáveis"de controlo (registos $t0 e $t1).
Se o caractere anterior não for número o programa volta para o `loop` e se a pilha estiver cheia, o programa mostra a mensagem "*Stack is full*".

6.8 Função barraN

A função `barraN` funciona de forma semelhante à função `espaco`, mas em vez de voltar ao `loop` caso o caractere anterior não seja um número, o programa dá print (através da função `print`) e depois volta para o ciclo.

6.9 Função print

Esta função começa por guardar em `$s0` o endereço da última posição da pilha (onde será colocado o próximo valor). Depois disso, guarda o endereço da pilha em `$s1` (endereço do primeiro elemento da pilha).
Depois disso dá print da *string "Stack:"* e verifica se a stack está vazia ou não. Se estiver vazia, o programa mostra a mensagem "(empty)". Se não estiver vazia, o programa mostra os elementos da pilha começando no primeiro (`$s1`) enquanto este, incrementação após incrementação, continue a ser menor que `$s0`, mostrando assim a pilha, deixando o primeiro elemento em cima, e o último elemento em baixo.

6.10 Função sair

Esta é a função que termina o programa. Apenas é chamada (pela função `off`) se a input for `"off"`.
O programa mostra a mensagem *"Bye!"* e termina colocando o valor `10` em `$v0` (`li $v0, 10`), seguido de `syscall`.

7 Observações

Esta calculadora foi desenvolvida utilizando o sistema operativo *Windows 10*. É possível que, quando corrida em *Linux*, se verifique alguns erros em relação aos caracteres e à forma como as *strings* são analisadas.
Quando a *input* introduzida é apenas uma instrução seguida de um '\n', é lido o caractere seguinte (caractere nulo). Se as linhas 63 e 64 não estivessem presentes, o programa apresentaria a mensagem de *"Unkown command. Type 'help' for available commands"*.