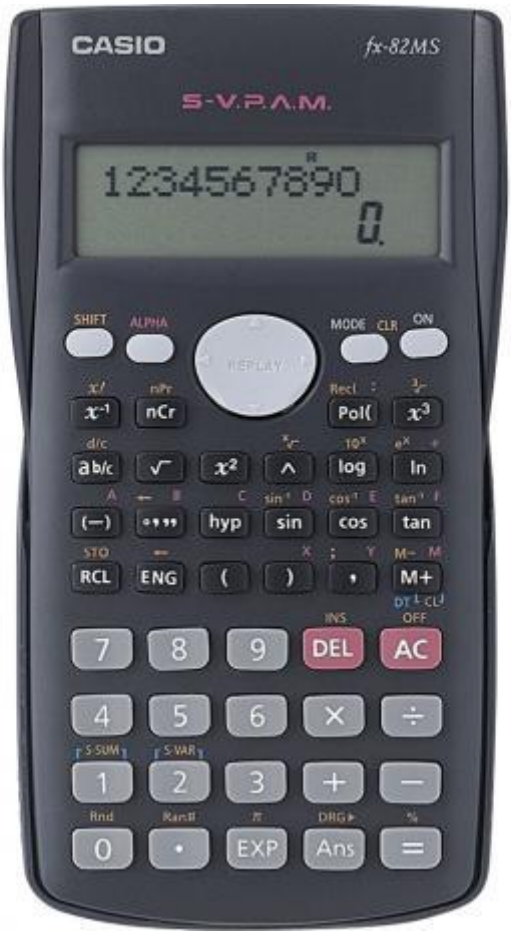
UNIVERSIDADE DE ÉVORA

Relatório

Calculadora em Assembly



Trabalho elaborado por:

Ruben Teimas nº 39868

Pedro Claudino nº39870

Índice

Introdução

Neste trabalho foi-nos proposto criar uma calculadora usando a linguagem assembly no simulador Mars Mips. A calculadora deverá realizar operações básicas com números inteiros e mais algumas funcionalidades.

Desenvolvimento

Começámos por definir as variáveis necessárias que queríamos guardar e de seguida definimos a função “main” onde estrará o corpo da calculadora.

```
.data
erro: .asciiz "Erro. Não pode dividir por 0!"
stack: .asciiz "Stack\n"
traço: .asciiz "\n->"
input: .space 256
.text
```

Labels

- Main: A label “main” recebe o input do utilizador com um máximo de 256 caracteres

```
main:
        li $v0, 4
        la $a0, traço
        syscall

        li $v0, 8
        la $a0, input
        li $a1, 256
        syscall

        addi $t3, $zero, 32
        addi $t4, $zero, 47
        addi $t6, $zero, 99
        addi $t7, $zero, 48
```

- Search: Com a label “search” iremos verificar se o input do utilizador é um número, caso seja irá para a função acumula. Se for uma operação irá saltar para a função escolha. Caso não seja um número, uma operação ou um espaço o programa termina.

```
search:
        lb $t0, 0($a0)
        addi $a0, $a0, 1
        lb $t1   , 0($a0)
        beq $t0, $t3, search
        nop

        slt $a2, $t4, $t0
        beq $a2, $zero, escolha
        nop

        bge $t0, $t6, escolha
        nop

        slt $a3, $t1, $t7
        beq $a3, $zero, acumula
        nop

        addi $sp, $sp, -4
        addi $t0, $t0, -48
        sw $t0, 0($sp)
        j search
        nop
```

- Escolha: Com a label “escolha” que irá escolher qual a operação dependendo do caracter introduzido pelo utilizador.

```
escolha:
        addi $t5, $zero, 43
        beq $t0, $t5, soma
        nop

        addi $t5, $zero, 45
        beq $t0, $t5, subtrai
        nop

        addi $t5, $zero, 42
        beq $t0, $t5, multiplica
        nop

        addi $t5, $zero, 47
        beq $t0, $t5, divide
        nop

        addi $t5, $zero, 99
        beq $t0, $t5, clear
        nop

        addi $t5, $zero, 100
        beq $t0, $t5, D
        nop

        addi $t5, $zero, 110
```

```
beq $t0, $t5, negate
nop

addi $t5, $zero, 115
beq $t0, $t5, swap
nop

addi $t5, $zero, 111
beq $t0, $t5, off
nop

beq $zero, $zero, resultado
nop
```

- D: A label “D” é chamada caso no input exista o char ‘d’ para escolher se realizará a operação “dup” ou delete.

```
D:
    lb $t0, 0($a0)

    addi $t5, $zero, 101
    beq $t0, $t5, delete
    nop

    addi $t5, $zero, 117
    beq $t0, $t5, duplica
    nop
```

- Soma/Subtrai/Multiplica/Divide: Nesta função (“soma”) iremos somar dois números utilizando a instrução “add” e depois irá voltar para a função “search” para verificar se há mais operações isto repete se nas funções “subtrai”, “multiplica”, “divide”.

```
soma:
        lw $a1, 0($sp)
        lw $a2, 4($sp)
        add $v0, $a1, $a2
        addi $sp, $sp, 4
        sw $v0, 0($sp)
        j search
        nop
```

- Clear: A label “clear” permite limpar a stack utilizando um ciclo que termina quando a stack estiver vazia.

```
clear:
        addi $sp, $sp, 4
        lw $a1, 0($sp)
        beq $a1, $zero, search
        nop
        j clear
        nop
```

- Divide: Na função “divide” iremos efetuar a divisão e verificar se o número está a ser divido por zero caso esteja irá mostrar uma mensagem de erro.

```
divide:
        lw $a1, 4($sp)
        lw $a2, 0($sp)
        beq $a2, $zero, ERRO
        div $a1, $a2
```

```
mflo $v0
mfhi $a3
addi $sp, $sp, 4
sw $v0, 0($sp)
j search
nop
```

- Erro: a label ERRO mostra a mensagem de erro caso o numero esteja a ser dividido por 0.

```
ERRO:
    li $v0, 4
    la $a0, erro
    syscall
    j main
    nop
```

- Delete: De seguida a label “delete” que irá servir para apagar o último número introduzido pelo o utilizador.

```
delete:
    addi $sp, $sp 4
    j search
    nop
```

- Duplica: Com a label “duplica” iremos duplicar o último número introduzido pelo o utilizador.

```
duplica:
        lw $a1, 0($sp)
        move $v0, $a1
        addi $sp, $sp, -4
        sw $v0, 0($sp)
        j search
        nop
```

- Negate: A label “negate” altera o sinal de um número para negativo.

```
negate:
        lw $a1, 0($sp)
        sub $v0, $zero, $a1
        sw $v0, 0($sp)
        j search
        nop
```

- Swap: Na label “swap” iremos trocar a posição dos dois últimos números introduzidos pelo o utilizador.

```
swap:
        lw $a1, 0($sp)
        move $at, $a1
        lw $a2, 4($sp)
        sw $a2, 0($sp)
        sw $at, 4($sp)
        j search
        nop
```

- Resultado: Com a label “resultado” iremos mostrar o último número guardado na stack.

```
resultado:
        lw $a0, 0($sp)
        li $v0, 1
        la $a0, ($a0)
        syscall
        j main
        nop
```

- Acumula: A label “acumula” se o char seguinte ao que está a ser lido é um algarismo, caso seja multiplica por dez e soma o anterior.

```
acumula:
        addi $t0, $t0, -48
        addi $t1 $t1, -48
        mulo $t9, $t0, 10
        add $v0, $t9, $t1
        add $sp, $sp, -4
        sw $v0, 0($sp)
        addi $a0, $a0, 1
        j search
        nop
```

- Off: A label “off” simplesmente desliga a calculadora.

```
off:
        li $v0, 10
        syscall
```

Conclusão

O trabalho foi parcialmente realizado com sucesso, dado que conseguimos realizar tudo o que nos foi pedido à exceção do display da stack.