CAPA

**1. Introdução**

As Linguagens Assembly são uma das linguagens de mais baixo nível, estando somente acima da linguagem binária. Esta linguagem possuí o propósito de traduzir o código em linguagem de montagem para a linguagem de máquina de acordo com a seu respectiva família de arquitetura do processador utilizado.

Tal linguagem é classificada em duas categorias: CISC (Complex

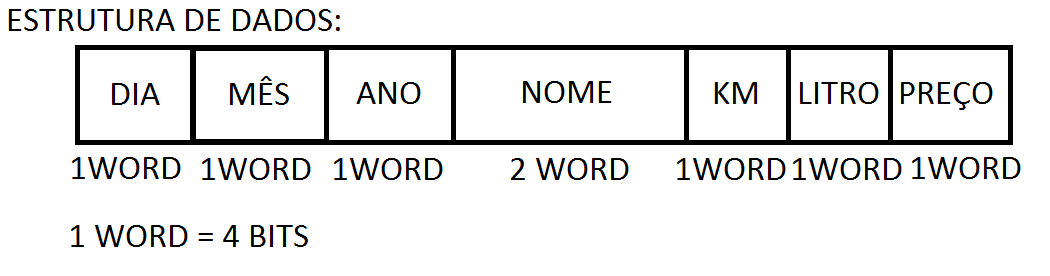
Instruction Set Computer, ou Conjunto Complexo de Instruções de Computador) e RISC (Reduced Instruction Set Computer, Conjunto Reduzido de Instruções de Computador). A linguagem MIPS é uma linguagem RISC, a qual foi elaborada visando um conjunto de instruções simples, tornando desnecessária a presença de um interpretador, o que faz com que as instruções sejam executadas diretamente pelo hardware. Portanto, pode-se dizer que as arquiteturas RISC são projetadas com o fim de fazer com que complexidade se encontre no compilador, e não no hardware ou no conjunto de instruções. Esta arquitetura tem como objetivo obter uma maior eficiência em relação ao tempo de execução das instruções no hardware. Além disso, devido à sua simplicidade, a arquitetura RISC apresenta um circuito interno mais enxuto, e, portanto, custo reduzido em relação a arquitetura CISC.

Com isso, foi elaborado um projeto através da linguagem assembly do MIPS, o qual consiste em um programa para controle do consumo de combustível de um automóvel, no qual é possível cadastrar, excluir e exibir os abastecimentos, exibir consumo médio e exibir o preço médio do combustível. Tal projeto foi elaborado com o auxílio de editores de texto e dos ambientes de simulação QtSpim e MARS, os quais são capazes de compilar e executar instruções da arquitetura MIPS32 (32 bits),

uma das inúmeras variantes da linguagem.

**2. Especificação**

**2.1 Detalhes de Projeto**

****

**2.2 Detalhes de Implementação**

Escolhemos esta estrutura, pois além de ser uma forma simples para se locomover na pilha em MIPS e a quantidade de bites para cada parâmetro foi feito de acordo com o desejado na descrição do projeto. Utilizamos também offsets para facilitar o entendimento do código (designar qual parte da estrutura representa o parâmetro desejado).

**3. Resultados**

**3.1 Testes Realizados**

Primeiramente testamos a opção 1 do menu (Cadastrar Abastecimento) onde cadastramos 3 abastecimentos.

1° cadastro

Data: 12/12/12

Nome do posto: IPIRANGA

Quilometragem: 45 km

Quantidade de combustível: 30 L

Preço: R$ 4,55

2° cadastro

Data: 13/12/12

Nome do posto: SHELL

Quilometragem: 45 km

Quantidade de combustível: 10 L

Preço: R$ 4,69

3° cadastro

Data: 11/12/12

Nome do posto: IPIRANGA

Quilometragem: 30 km

Quantidade de combustível: 10 L

Preço: R$ 4,89

Todos os cadastros foram feitos com sucesso.

Teste 2 no menu escolhemos uma opção invalida, o que retorna uma mensagem de opção invalida e retorna ao menu.

No terceiro teste escolhemos a opção 2 do menu (Exibe Abastecimento), que exibe os abastecimentos cadastrados ordenados por data.

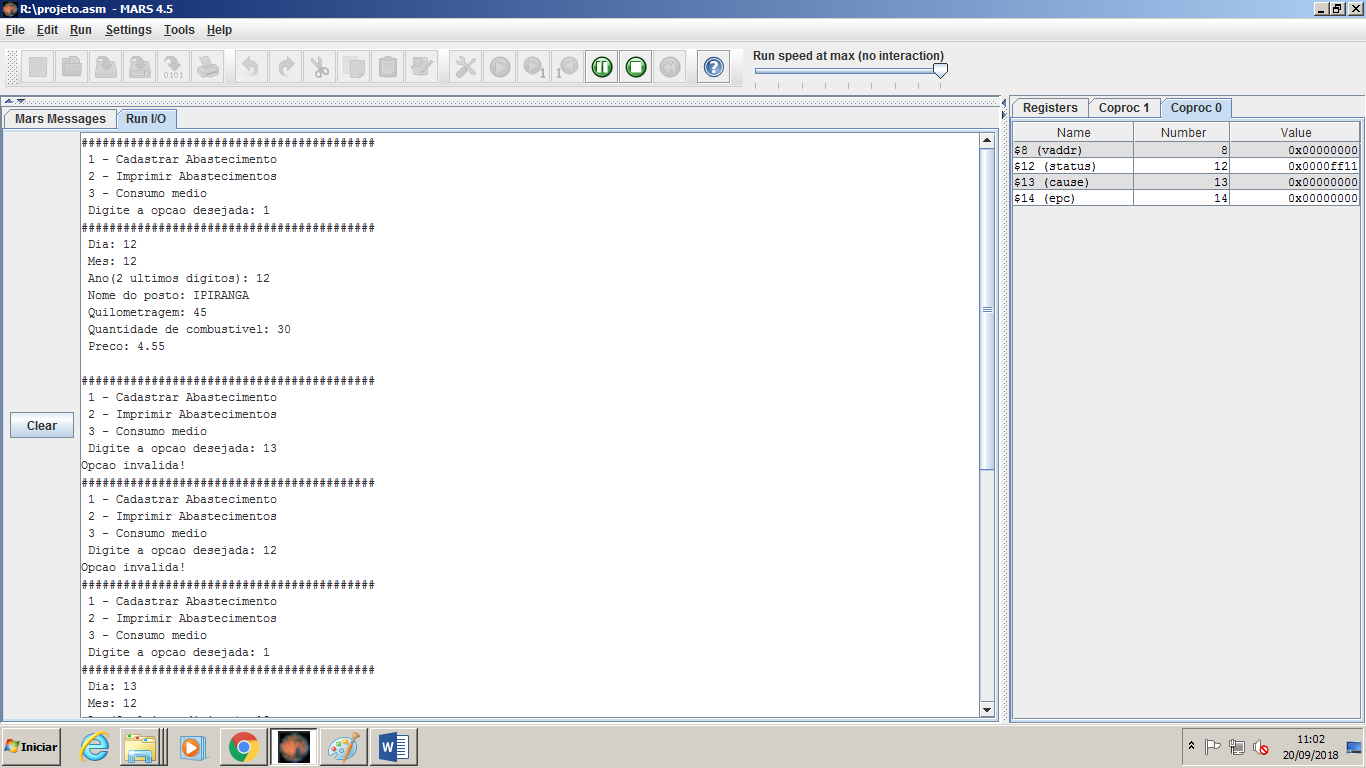
No quarto teste escolhemos a opção 3 do menu (Exclui Abastecimento), que exclui com sucesso os dados do abastecimento realizado em uma data informada pelo usuário.

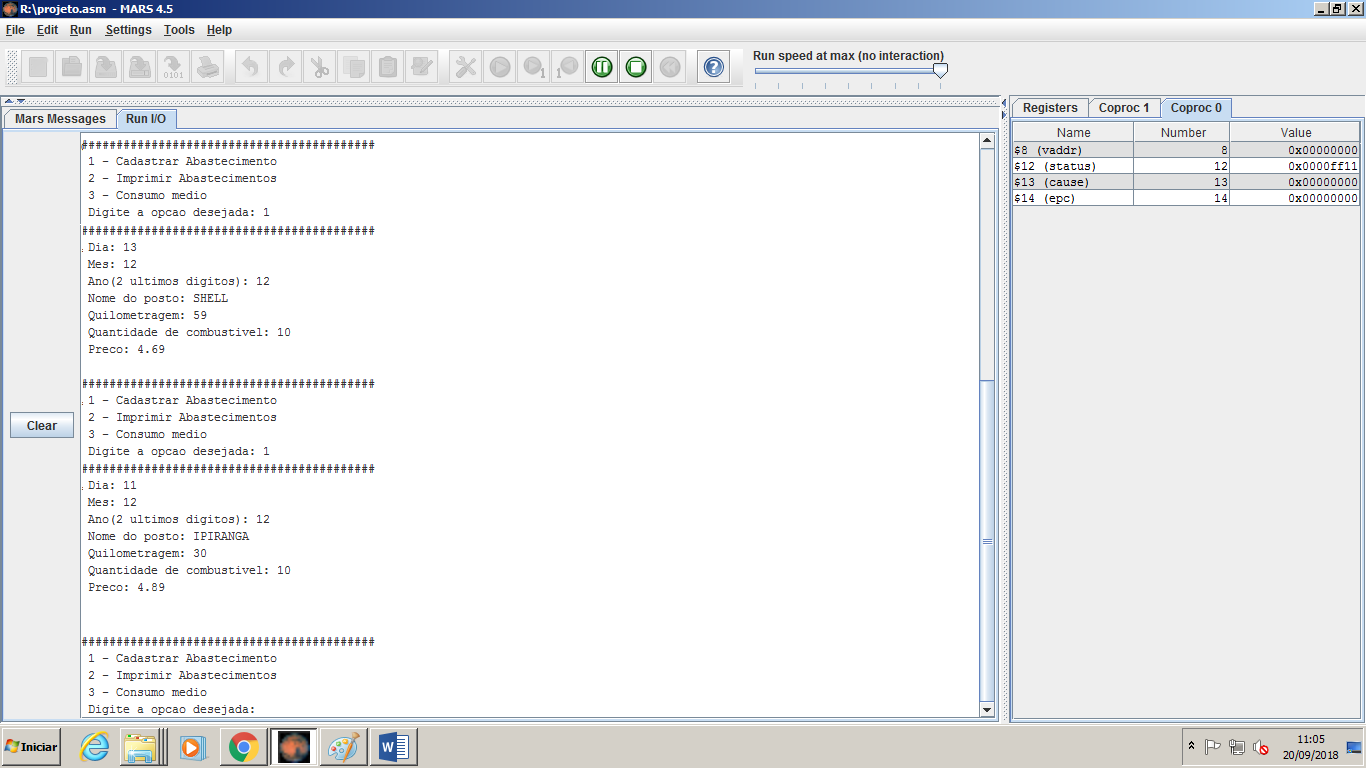
No quinto teste escolhemos a opção 4 (Consumo Médio) onde exibimos com sucesso o cálculo do consumo médio do veículo (km/ litros) de acordo com os dados de abastecimento cadastrados até o momento.

No sexto teste escolhemos a opção 5 (Preço Médio) onde exibimos o cálculo do preço médio do litro de combustível por posto de acordo com os dados de abastecimento cadastrados até o momento, exibindo com sucesso o nome do posto e o preço médio em ordem decrescente de preço.

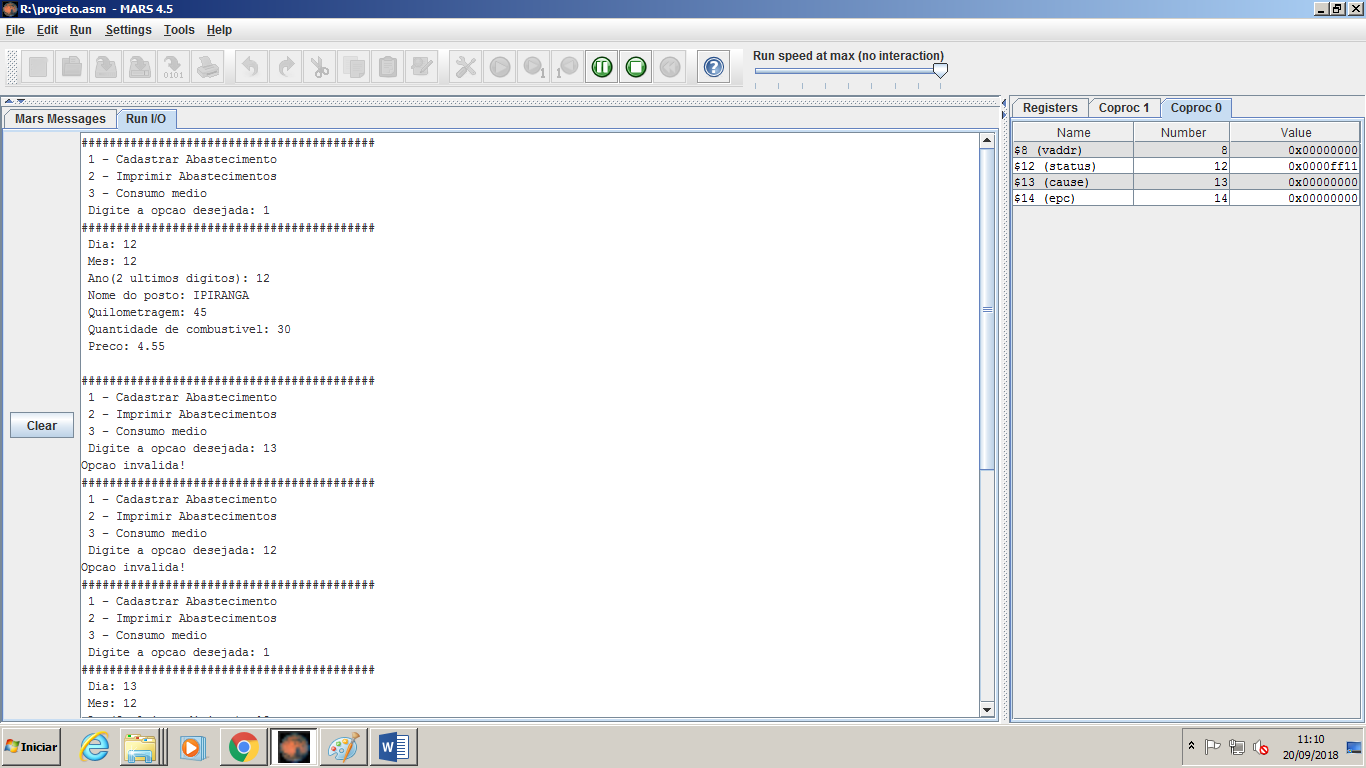
**3.2 Resultados e Discussões**

Teste 1

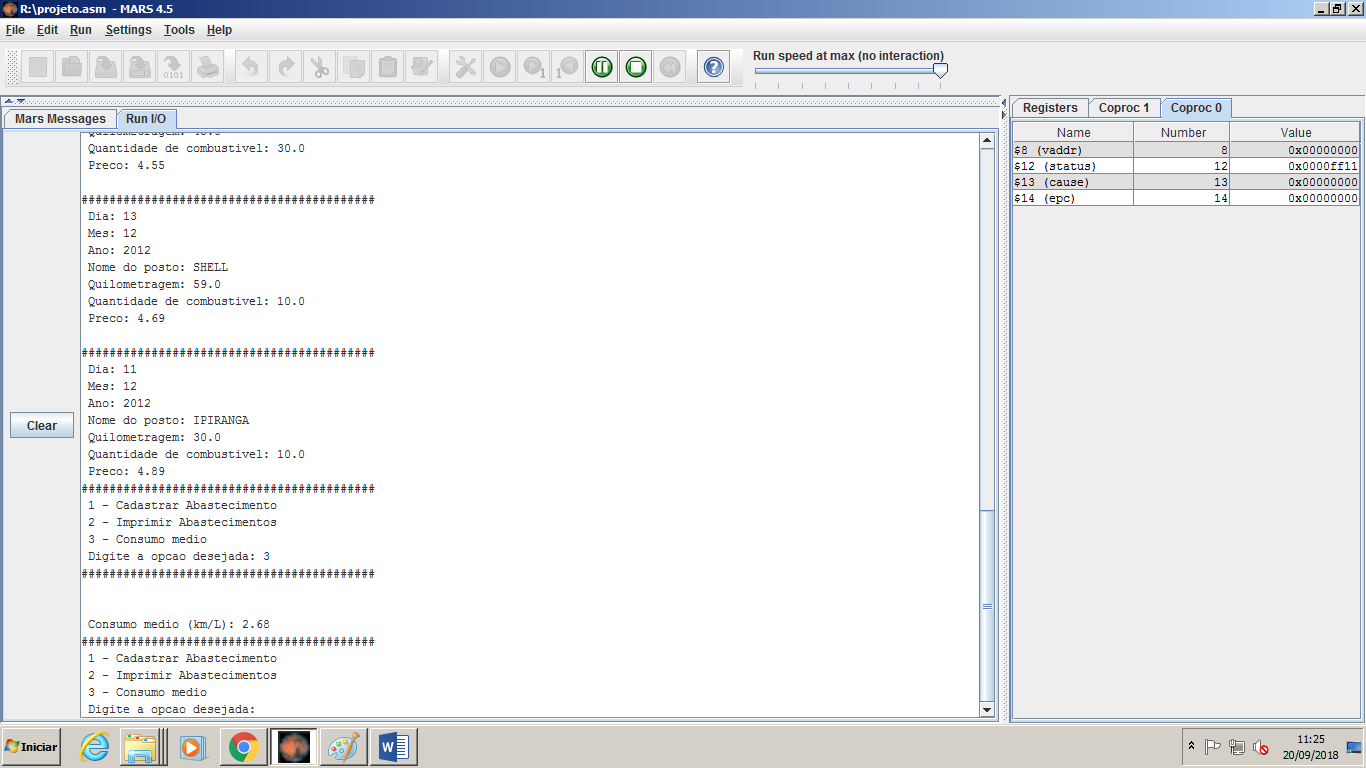


****

Teste 2



Teste 5



**4. Bibliografia**

.data

.align 4

ABASTECIMENTOS: .space 4000 #10(w)\*4(B)\*100(armazenamentos)

ULTIMO\_ABASTECIMENTO: .word 0

ABASTECIMENTO\_POINTER: .word 0

ABASTECIMENTO\_COUNT: .word 0

# ESTRUTURA DE DADOS:

# [ dia | mes | ano | nome | km | litro | preço ]

# ( 01w | 01w | 01w | 16b | 01w | 01w | 01w )

# ----------------------------------------------------

# OFFSETS INTERNOS: (em bytes)

.eqv DIA\_OFFSET 0 # dia: 0

.eqv MES\_OFFSET 4 # mes: 4

.eqv ANO\_OFFSET 8 # ano: 8

.eqv NOME\_OFFSET 12 # nome: 12

.eqv KM\_OFFSET 28 # km: 28

.eqv LITRO\_OFFSET 32 # litro: 32

.eqv PRECO\_OFFSET 36 # preço: 36

# ======================================================

MENU: .asciiz " 1 - Cadastrar Abastecimento\n 2 - Imprimir Abastecimentos\n 3 - Consumo medio\n Digite a opcao desejada: "

TXT\_INVALIDO: .asciiz "Opcao invalida!\n"

PULA\_LINHA: .asciiz "\n"

TXT\_DIA: .asciiz " Dia: "

TXT\_MES: .asciiz " Mes: "

TXT\_ANO: .asciiz " Ano(2 ultimos digitos): "

TXT\_ANO\_EXIBIR: .asciiz " Ano: 20"

TXT\_NOME: .asciiz " Nome do posto: "

TXT\_KM: .asciiz " Quilometragem: "

TXT\_COMBUSTIVEL: .asciiz " Quantidade de combustivel: "

TXT\_CONSUMO\_MEDIO: .asciiz " Consumo medio (km/L): "

TXT\_PRECO: .asciiz " Preco: "

DIVISOR: .asciiz "##########################################"

.text

.globl main

main:

printaMenu:

addi $v0, $zero, 4

la $a0, DIVISOR

syscall

jal println

addi $v0, $zero, 4 # para printar uma string colocar o codigo 4 em v0

la $a0, MENU # colocar o endereco da mensagem em a0

syscall

addi $v0, $zero, 5 # para receber um inteiro colocar codigo 5

syscall

addi $s0, $v0, 0 # o inteiro digitado fica salvo em v0 guardar em s0

li $s1, 1 # colocar 1 em s1 para comparar se digitou opcao 1

beq $s1, $s0, cadastraAbastecimentoPonte

li $s1, 2 # colocar 2 em s1 para comparar se digitou opcao 2

beq $s1, $s0, imprimirTudo

li $s1, 3 # colocar 2 em s1 para comparar se digitou opcao 2

beq $s1, $s0, consumoMedio

addi $v0, $zero, 4 # para printar uma string colocar o codigo 4 em v0

la $a0, TXT\_INVALIDO # colocar o endereço da mensagem em a0

syscall

j voltaMenu

cadastraAbastecimentoPonte: j cadastraAbastecimento

############################################################################

cadastraAbastecimento:

addi $v0, $zero, 4

la $a0, DIVISOR

syscall

la $t1, ABASTECIMENTOS # carrega endereço de ABASTECIMENTOS em t1

lw $t0, ULTIMO\_ABASTECIMENTO #valor 0~99

mul $t0, $t0, 40

add $t1, $t1, $t0 #endereço real do ultimo abastecimento em $t1

# $t1 contem endereço do abastecimento em si (FIXO)

jal println

#DIA

addi $v0, $zero, 4

la $a0, TXT\_DIA

syscall

addi $v0, $zero, 5 # para receber um inteiro colocar codigo 5

syscall # value in $v0

sw $v0, DIA\_OFFSET($t1)

#MES

addi $v0, $zero, 4

la $a0, TXT\_MES

syscall

addi $v0, $zero, 5 # para receber um inteiro colocar codigo 5

syscall # value in $v0

sw $v0, MES\_OFFSET($t1)

#ANO

addi $v0, $zero, 4

la $a0, TXT\_ANO

syscall

addi $v0, $zero, 5 # para receber um inteiro colocar codigo 5

syscall # value in $v0

sw $v0, ANO\_OFFSET($t1)

#NOME

addi $v0, $zero, 4

la $a0, TXT\_NOME

syscall

addi $v0,$zero,8

la $a0,NOME\_OFFSET($t1)

addi $a1,$zero,16

syscall

#KM

addi $v0, $zero, 4

la $a0, TXT\_KM

syscall

addi $v0, $zero, 6 # para receber um float 6

syscall # value in $f0

s.s $f0, KM\_OFFSET($t1)

#QUANTIDADE COMBUSTIVEL

addi $v0, $zero, 4

la $a0, TXT\_COMBUSTIVEL

syscall

addi $v0, $zero, 6 # para receber um float 6

syscall # value in $f0

s.s $f0, LITRO\_OFFSET($t1)

#PRECO

addi $v0, $zero, 4

la $a0, TXT\_PRECO

syscall

addi $v0, $zero, 6 # para receber um float 6

syscall # value in $f0

s.s $f0, PRECO\_OFFSET($t1)

jal println

#ULTIMO\_ABASTECIMENTO++

lw $t0, ULTIMO\_ABASTECIMENTO

addi $t0, $t0, 1

sw $t0, ULTIMO\_ABASTECIMENTO

#ABASTECIMENTO\_COUNT = ULTIMO\_ABASTECIMENTO

sw $t0, ABASTECIMENTO\_COUNT

j voltaMenu

######################## VOLTA PARA O MENU #########################

voltaMenu:

j printaMenu

############################### PRINTA NOVA LINHA #########################

println:

addi $v0, $zero, 4

la $a0, PULA\_LINHA

syscall

jr $ra

############################### IMPRIME UM ABASTECIMENTO #########################

imprimirAbastecimento:

addi $v0, $zero, 4

la $a0, DIVISOR

syscall

addi $sp, $sp, -4 # $sp = $sp - 4 configure offset

sw $ra, 0($sp) # save on stack

jal println

#<------------DIA------------>

addi $v0, $zero, 4

la $a0, TXT\_DIA

syscall

lw $a0, DIA\_OFFSET($t1) #carrega valor do teclado no $a0

li $v0, 1 # get print integers

syscall

jal println # jump to println and save position to $ra

#<------------DIA------------>

#<------------MES------------>

addi $v0, $zero, 4

la $a0, TXT\_MES

syscall

lw $a0, MES\_OFFSET($t1) #carrega valor do teclado no $a0

li $v0, 1 # get print integers

syscall

jal println # jump to println and save position to $ra

#<^^^^^^^^^^^^MES^^^^^^^^^^^^>

#<------------ANO------------>

addi $v0, $zero, 4

la $a0, TXT\_ANO\_EXIBIR

syscall

lw $a0, ANO\_OFFSET($t1) #carrega valor do teclado no $a0

li $v0, 1 # get print integers

syscall

jal println # jump to println and save position to $ra

#<^^^^^^^^^^^^ANO^^^^^^^^^^^^>

#<------------NOME------------>

addi $v0, $zero, 4

la $a0, TXT\_NOME

syscall

la $a0,NOME\_OFFSET($t1)

li $v0, 4 # print string

syscall

# jal println # jump to println and save position to $ra

#<^^^^^^^^^^^^NOME^^^^^^^^^^^^>

#<------------KM------------>

addi $v0, $zero, 4

la $a0, TXT\_KM

syscall

l.s $f12, KM\_OFFSET($t1)

li $v0, 2 # get print float

syscall

jal println # jump to println and save position to $ra

#<^^^^^^^^^^^^KM^^^^^^^^^^^^>

#<------------LITRO------------>

addi $v0, $zero, 4

la $a0, TXT\_COMBUSTIVEL

syscall

l.s $f12, LITRO\_OFFSET($t1)

li $v0, 2 # get print float

syscall

jal println # jump to println and save position to $ra

#<^^^^^^^^^^^^LITRO^^^^^^^^^^^^>

#<------------PRECO------------>

addi $v0, $zero, 4

la $a0, TXT\_PRECO

syscall

l.s $f12, PRECO\_OFFSET($t1)

li $v0, 2 # get print float

syscall

jal println # jump to println and save position to $ra'

#<------------PRECO------------>

lw $ra, 0($sp) # restore stack

jr $ra # jump to $ra

############################### IMPRIME OS ABASTECIMENTO NA PILHA #########################

imprimirTudo:

lw $t0, ULTIMO\_ABASTECIMENTO # valor 0~99

la $t1, ABASTECIMENTOS # carrega endereço de ABASTECIMENTOS em t1

li $t2, 0 # $t2 =0

beq $t0, $zero, exit # if $t0 == $t1 then exit

loop:

jal imprimirAbastecimento

addi $t1, $t1, 40

add $a0, $t1, $zero # $a0 = $t1 +zerot2

addi $t2, $t2, 1 # $t2 = $t2 + 1

beq $t0, $t2, exit # if $t0 == $t1 then exit

jal println # jump to println and save position to $ra'

j loop # jump to target

############################### EXIBE CONSUMO MEDIO #########################

consumoMedio:

addi $v0, $zero, 4

la $a0, DIVISOR

syscall

jal println # jump to println and save position to $ra'

lw $t0, ULTIMO\_ABASTECIMENTO # valor 0~99

la $t1, ABASTECIMENTOS # carrega endereço de ABASTECIMENTOS em t1

li $t2, 0 # $t2 =0

mtc1 $zero, $f0 # inicializa o registrador com 0

mtc1 $zero, $f1 # inicializa o registrador com 0

mtc1 $zero, $f3 # inicializa o registrador com 0

beq $t0, $zero, exit # if $t0 == $t1 then exit

loop\_consumo:

l.s $f12, KM\_OFFSET($t1) # carrega o valor da memoria

add.s $f0,$f0,$f12 # add de floats

l.s $f12, LITRO\_OFFSET($t1) # carrega o valor da memoria

add.s $f1,$f1,$f12 # add de floats

addi $t1, $t1, 40 # anda na memoria para percorrer o vetor

add $a0, $t1, $zero # $a0 = $t1 +zerot2

addi $t2, $t2, 1 # $t2 = $t2 + 1

beq $t0, $t2, exit\_consumo # if $t0 == $t1 then exit

jal println # jump to println and save position to $ra'

j loop\_consumo # jump to loop\_consumo

exit\_consumo:

div.s $f3, $f0, $f1 # divisao dos floats

addi $v0, $zero, 4

la $a0, TXT\_CONSUMO\_MEDIO

syscall

mov.s $f12,$f3 # movimenta o float para ser impresso

li $v0, 2 # get print float

syscall

jal println # jump to println and save position to $ra

j voltaMenu # jump to voltaMenu

exit:

j voltaMenu # jump to voltaMenu