Fundamentos de Arquitetura de Computadores

Trabalho 02

Professor: Tiago Alves
Turma: A
Alunos e matrículas:
Gabriela Barrozo Guedes 16/0121612
Vitor Leal dos Santos 16/014375

1.Introdução

Para este trabalho foi pedido a implementação de um programa na linguagem de montagem assembly mips que calcule a exponenciação modular a partir dos números fornecidos. Dessa forma o usuário deve fornecer a base, o expoente e o módulo e o programa deve fazer o cálculo e retornar o resultado caso o módulo for primo, se o módulo não for primo o programa deverá imprimir um erro.

2. Implementação

2.1 Ferramentas utilizadas

O programa foi construído utilizando um sistema operacional Linux, mais especificamente, o Ubuntu 17.10. Foi utilizado também o Mars 4.5 como ambiente de desenvolvimento durante sua implementação.

2.2 Código

No início do código salvamos as strings que serão utilizadas e já pulamos para a função ***MAIN*** do programa.

```
1  .data
2          comando: .asciiz "Escreva as variaveis para base, expoente e modulo"
3          base: .asciiz "base: "
4          expoente: .asciiz "expoente: "
5          modulo: .asciiz "modulo: "
6          primo: .asciiz "eh primo"
7          erro: .asciiz "O modulo nao eh primo"
8          saida1: .asciiz "A exponencial modular "
9          saida2: .asciiz " elevado a "
10         saida3: .asciiz " (mod "
11         saida4: .asciiz ") eh "
12         saida5: .ascii "."
13 .text
14
15 j MAIN
16
```

Primeiramente o programa deve imprimir as instruções para orientar o usuário. Logo após as instruções iniciais, o programa pedirá a base, a exponencial e o módulo utilizando a função ***le_inteiro*** e salvando os números nas variáveis ***$t1***,***$t2*** e ***$t3*** respectivamente.

```
111 MAIN:
112
113         #Comando inicial
114         li $v0, 4
115         la $a0, comando
116         syscall #Imprime a label
117
118         li $a0, 0xA
119         li $v0, 0xB
120         syscall ##imprime a quebra de linha
121
122         #BASE
123         li $v0, 4
124         la $a0, base
125         syscall #Imprime a label
126
127         jal le_inteiro #salva o número
128         move $t1, $t0 #$t1 = BASE
```

```
130         #EXPOENTE
131         li $v0, 4
132         la $a0, expoente
133         syscall
134
135         jal le_inteiro
136         move $t2, $t0 #$t2 = EXPOENTE
137
138         #MODULO
139         li $v0, 4
140         la $a0, modulo
141         syscall
142
143         jal le_inteiro
144         move $t3, $t0 #$t3 = MODULO (possivel primo)
```

```
16
17 le_inteiro:
18         li $v0, 5 #Seleciona tipo de variavel a ser lido (INTEIRO = 5)
19         syscall
20
21         move $t0, $v0
22         jr $ra
23
```

Em seguida verificamos se o número é primo pela função ***eh_primo***, se o número for primo o valor de ***$s0*** será 1, caso contrario ***$s0*** continuará como 0.

```
145
146         jal eh_primo
147
```

Para verificar se o módulo é primo e feito divisões de 1 até o valor do módulo, e contando quantas dessas divisões possuem resto = 0. Dessa forma, os números que possuem exatamente 2 divisões com resto zero são primos.

```
24 eh_primo:
25         add $t5, $0, $0 #Variavel para contar quantas divisões tem o número
26         li $t6, 1 #Variavel i do for
27         j FOR # Chama o for
28
29
30         IF:
31                 addi $t5, $t5, 1 #Adiciona 1 na variavel que conta as divisões
32                 j volta_do_if #volta para o for
33
34         IF2:
35                 li $s0,  1 #Coloca 1 na variavel $s0 se for primo
36                 j volta_do_if2 #volta
37
38         FOR:
39                 div $t3, $t6 #Divide o provavel primo por i
40                 mfhi $t7 #armazena o resto
41                 beqz $t7, IF #Se resto=0, entra no if
42                 volta_do_if:
43                 addi $t6, $t6, 1 # i++
44                 ble $t6, $t3, FOR # Se i <= provavel primo, continua o for
45
46         add $s0, $0, $0 #Inicia a variavel de retorno como 0
47         beq  $t5, 2, IF2 # Se tiver 2 divisores entra no if2
48         volta_do_if2:
49         jr $ra #volta pra main
50
```

Após verificar se o número é primo, se ***$s0*** for igual a zero(ou seja, se o número não for primo), é chamada a função ***imprime_erro*** que imprime na tela *“O modulo nao eh primo”* e pula para o final do programa.

```
147
148         beqz $s0, imprime_erro
149
```

```
63
64  imprime_erro:
65          li $v0, 4
66          la $a0, erro
67          syscall
68          j END
69
70  imprime_saida:
```

Como já verificamos que o módulo é primo, a próxima função chamada é a ***calc_exp*** para calcular a exponencial e salvar o resultado em ***$s1***.

```
149
150         jal calc_exp
151         jal imprime saida
```

```
50
51  calc_exp:
52          add $s1, $0, $t1 # resultado = base
53          li $t6, 1 #variavel i do for começando com 1
54
55          FOR2:
56                  mul $s1, $s1, $t1
57                  addi $t6, $t6, 1 #i++
58                  blt $t6, $t2, FOR2 #Continua o for até i < expoente
59
60          div $s1, $t3 #Divide o resultado pelo modulo
61          mfhi $s1 #armazena o resto na variavel de resultado
62          jr $ra
63
```

Com o resultado armazenado em ***$s1***, chamamos função ***imprime_saida*** que imprime o resultado no seguinte formato *"A exponencial modular AA elevado a BB (mod PP) eh ZZ."*.

```
150         jal calc_exp
151         jal imprime_saida
152
153 END:
```

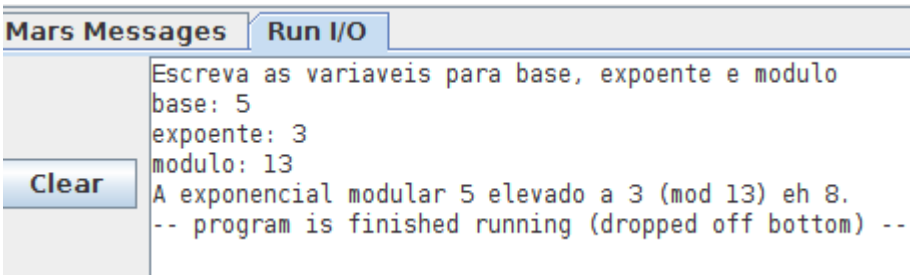
```
69
70  imprime_saida:
71          li $v0, 4
72          la $a0, saida1
73          syscall
74
75          li $v0, 1
76          move $a0, $t1
77          syscall # Imprime a BASE
78
79          li $v0, 4
80          la $a0, saida2
81          syscall
82
83          li $v0, 1
84          move $a0, $t2
85          syscall # Imprime O EXPONECIAL
```

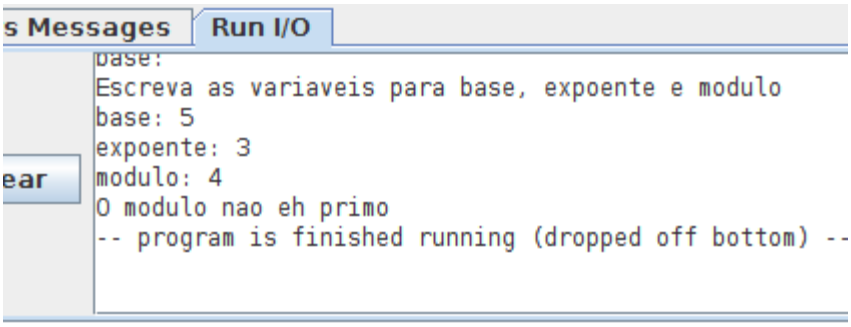
```
87          li $v0, 4
88          la $a0, saida3
89          syscall
90
91          li $v0, 1
92          move $a0, $t3
93          syscall # Imprime o MODULO
94
95          li $v0, 4
96          la $a0, saida4
97          syscall
98
99          li $v0, 1
100         move $a0, $s1
101         syscall # Imprime o RESULTADO
102
103         li $v0, 4
104         la $a0, saida5
105         syscall
106
107         jr $ra
```

3. Instruções de uso

3.1 Telas e comandos

O usuário deve compilar o código e digitar os números para base, exponencial e módulo quando pedido pelo programa e ver o resultado.





3.2 Limitações conhecidas

O programa precisa ser executado a partir do Mars 4.5