Função de Raiz Quadrada usando o Método Newton-Raphson Recursivo

Organização e Arquitetura de Processadores

```
Eduarda Patricio (23111258-2) • Giovanna Castro (23111285-2) Naiumy dos Reis (23111738-3) • Yasmin Aguirre (23111329-1)
```

Algoritmo em alto nível

O programa abaixo contém a implementação do método Newton-Raphson de forma recursiva na linguagem Java.

```
import java.util.*;
public class altoNivel {
  // Método para calcular a raiz quadrada usando o método Newton-Raphson recursivamente
  public static int sqrt_nr(int x, int i) {
    int result = 0;
    // Caso base: se i for 0, retorna 1 e faz o primeiro cálculo do método
    if (i == 0) {
      result = 1;
    }
    // Caso contrário, faz o cálculo i-1 do método recursivamente
      result = (\operatorname{sqrt\_nr}(x, i - 1) + (x / \operatorname{sqrt\_nr}(x, i - 1))) / 2;
    }
    return result;
  }
  public static void main(String[] args) {
    Scanner in = new Scanner(System.in);
    int i = 0;
    int x = 0;
    System.out.println("\nPrograma de Raiz Quadrada - Newton-Raphson\r\n" +
         "Desenvolvedoras: Eduarda Patricio, Giovanna Castro, Naiumy dos Reis e Yasmin Aguirre");
    while (true) {
      System.out.println("\nDigite os parâmetros x e i para calcular sqrt_nr (x, i) ou -1 para abortar a execução");
      System.out.println("Digite o parâmetro x:");
      x = in.nextInt();
      if (x < 0) {
         break;
      System.out.println("Digite o parâmetro i:");
      i = in.nextInt();
```

```
if (i < 0) {
          break;
    }
    // Chama o método sqrt_nr e exibe o resultado
    int a = sqrt_nr(x, i);
    System.out.println("sqrt(" + x + ", " + i + ") = " + a);
    }
    in.close();
}</pre>
```

2. Algoritmo em baixo nível

O programa abaixo contém a implementação do método Newton-Raphson de forma recursiva em Assembly do MIPS.

```
.macro prtStr(%string)
                                      # Macro para printar strings
addi
            $sp, $sp, -8
                                      # Abre dois espacos na pilha
            $v0, 0($sp)
                                      # Guarda o valor que esta no registrador $v0 no topo da pilha
SW
            $a0, 4($sp)
                                      # Guarda o valor que esta no registrador $a0 na segunda posicao da pilha
sw
            $a0, %string
                                      # Carrega o endereco da string em $a0
la
li
            $v0, 4
                                      # Carrega a instrucao 4 (print_string) em $v0
                                      # Chama o sistema para executar a instrucao
syscall
                                      # Recupera o valor que esta na segunda posicao da pilha de volta ao $a0
lw
            $a0, 4($sp)
lw
            $v0, 0($sp)
                                      # Recupera o valor que esta no topo da pilha de volta ao $v0
                                      # Apaga o espaco aberto na pilha
addi
            $sp, $sp, 8
.end macro
.macro leInt(%int)
                                      # Macro para ler um valor inteiro
addi
            $sp, $sp, -4
                                      # Abre espaco na pilha para um int
            $v0, 0($sp)
                                      # Coloca o que esta no registrador $v0 no topo da pilha
SW
                                      # Carrega a instrucao 5 (read_int) em $v0
li
            $v0, 5
                                      # Chama o sistema para executar a instrucao
syscall
move
            %int, $v0
                                      # Move o valor de $v0 para o destino %int
lw
            $v0, 0($sp)
                                      # Recupera o valor que esta no topo da pilha de volta ao $v0
addi
            $sp, $sp, 4
                                      # Apaga o espaco aberto na pilha
.end_macro
.macro prtInt(%inteiro)
                                      # Macro para printar um valor inteiro
addi
            $sp, $sp, -8
                                      # Abre dois espacos na pilha
            $v0, 0($sp)
                                      # Guarda o valor que esta no registrador $v0 no topo da pilha
sw
                                      # Guarda o valor que esta no registrador $a0 na segunda posicao da pilha
sw
            $a0, 4($sp)
                                      # Carrega o endereco do inteiro em $a0
            $a0, %inteiro
lw
li $v0, 1
                                      # Carrega a instrucao 1 (print int) em $v0
                                      # Chama o sistema para executar a instrucao
syscall
lw
            $a0, 4($sp)
                                      # Recupera o valor que esta na segunda posicao da pilha de volta ao $a0
            $v0, 0($sp)
                                      # Recupera o valor que esta no topo da pilha de volta ao $v0
lw
```

```
.end_macro
.macro callRaiz()
                                       # Macro para chamar a funcao raiz
addi
            $sp, $sp, -12
                                       # Abre tres espacos na pilha
            $a0, 8($sp)
                                       # Guarda o valor que esta no registrador $a0 (x) na terceira posicao da pilha
sw
            $a1, 4($sp)
                                       # Guarda o valor que esta no registrador $a1 (i) na segunda posicao da pilha
sw
            $ra, 0($sp)
                                       # Guarda o valor que esta no registrador $r0 no topo da pilha
sw
                                       # Vai para raiz
jal raiz
            $ra, 0($sp)
lw
                                       # Recupera o valor que esta no topo da pilha de volta ao $ra
lw
            $a1, 4($sp)
                                       # Recupera o valor que esta na segunda posicao da pilha de volta ao $a1
            $a0, 8($sp)
                                       # Recupera o valor que esta na terceira posicao da pilha de volta ao $a0
lw
addi
            $sp, $sp, 12
                                       # Apaga o espaco aberto na pilha
.end_macro
.macro resultado()
                                       # Macro para printar o resultado
sw
            $v0, result
                                       # Salva em result o valor lido em $v0
prtStr(strRaizI)
                                       # Print("sqrt("
prtInt(x)
                                       # Print("sqrt(" + x
prtStr(strVirg)
                                       # Print("sqrt(" + x + ", "
prtInt(i)
                                       # Print("sqrt(" + x + ", " + i
                                       # Print("sqrt(" + x + ", " + i + ") = "
prtStr(strRaizF)
                                       # Print("sqrt(" + x + ", " + i + ") = " + result);
prtInt(result)
.end_macro
.macro exit()
                                       # Macro para sair do programa
li
            $v0, 10
                                       # Carrega a instrucao 10 (exit) em $v0
                                       # Chama o sistema para executar a instrucao
syscall
.end macro
.data
x: .space 4
i: .space 4
result: .space 4
strProgm: .asciiz "\nPrograma de Raiz - Newton-Raphson\nDesenvolvedoras: Eduarda Patricio, Giovanna Castro,
Naiumy dos Reis e Yasmin Aguirre"
strDigite: .asciiz "\n\nDigite os parametros x e i para calcular sqrt_nr(x, i) ou -1 para abortar a execucao\nDigite o
parametro x: "
strPrml: .asciiz "Digite o parametro i: "
strRaizI: .asciiz "sqrt("
strVirg: .asciiz ", "
strRaizF: .asciiz ") = "
.text
   .globl main
main:
prtStr(strProgm)
                                       # Usa o macro prtStr para printar a string de strProgm
```

Apaga o espaco aberto na pilha

addi

\$sp, \$sp, 8

loop:

prtStr(strDigite) # Usa o macro prtStr para printar a string de strDigite leInt(\$a0) # Usa o macro leInt para ler o valor inteiro de \$a0

sw \$a0, x # Salva em x o valor lido em \$a0

blt \$a0, \$zero, fim # Verifica se a entrada e negativa (x < 0)

prtStr(strPrmI) # Usa o macro prtStr para printar a string de strPrmI leInt(\$a1) # Usa o macro leInt para ler o valor inteiro de \$a1

sw \$a1, i # Salva em i o valor lido em \$a1

blt \$a1, \$zero, fim # Verifica se a entrada e negativa (i < 0)

callRaiz() # Chama o macro que faz a funcao de raiz quadrada

resultado() # Chama o macro que imprime o resultado

jal loop # Repete o programa enquanto a entrada for positiva

fim:

exit() # Chama o macro que encerra o programa

raiz:

 Iw
 \$a1, 4(\$sp)
 # Carrega o valor de \$a1 em i

 Iw
 \$a0, 8(\$sp)
 # Carrega o valor de \$a0 em x

 bgt
 \$a1, \$zero, recurs
 # Se (i > 0), vai para "recurs"

li \$v0, 1 # Define o resultado como 1 se (i == 0)

jr \$ra # Volta para o macro callRaiz()

recurs:

addiu \$a1, \$a1, -1 # Decrementa i (i--)

callRaiz() # Chama o macro que faz a funcao de raiz quadrada de novo

addiu \$a1, \$a1, 1 # Restaura o valor de i (i++)

div \$t0, \$a0, \$v0 $\# x / sqrt_nr(x, i-1)$

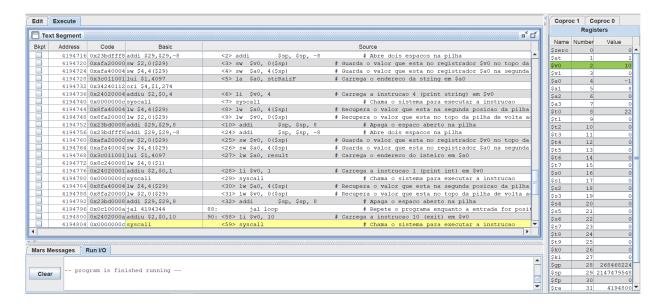
add \$v0, \$v0, \$t0 $$sqrt_nr(x, i-1) + (x / sqrt_nr(x, i-1))$ srl \$v0, \$v0, 1 $$sqrt_nr(x, i-1) + (x / sqrt_nr(x, i-1))) / 2$

jr \$ra # Volta para raiz

3. Capturas de tela do MARS

Screenshots do simulador MARS:

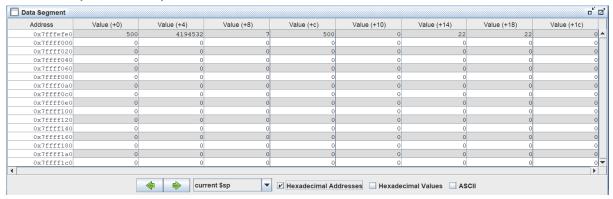
a) Área de código compilada:



b) Estado dos registradores ao final da execução:

Registers	Coproc 1 Coproc	: 0	
Name		Number	Value
\$ v 0		2	10
\$v1		3	(
\$a0		4	- :
\$a1		5	
\$a2		6	
\$a3		7	
\$t0		8	2.
\$t1		9	
\$t2		10	
\$t3		11	
\$t4		12	
\$t5		13	
\$t6		14	
\$t7		15	
\$s0		16	
\$s1		17	
\$s2		18	
\$s3		19	
\$s4		20	
\$s5		21	
\$s6		22	
\$s7		23	
\$t8		24	
\$t9		25	
\$k0		26	
\$k1		27	
\$gp		28	26846822
\$sp		29	214747954
\$fp		30	
Şra		31	419480
pc			419480
hi			1
lo			2

c) Área de pilha utilizada para a recursividade:



d) Exemplo de execução do programa:

