Sumário

1. Introdução:

A sequência de Fibonacci é uma sequência de números inteiros que começa por 0 ou 1 e cada elemento é o resultado da soma dos 2 elementos anteriores.

A razão áurea é um número irracional cujo valor é aproximadamente 1,61803398875 e possui uma representação decimal infinita.

É importante observar que a medida que a sequência avança, a divisão entre elementos consecutivos se aproxima ainda mais da razão áurea.

O objetivo deste exercício é calcular o elemento na posição 30 que deve ser armazenado no registrador $s1, calcular o elemento na posição 41 que deve ser armazenado no registrador $s2, calcular o elemento na posição 40 que deve ser armazenado no registrador $s3 e calcular a razão áurea dos elementos 41 e 40 que deve ser armazenada no registrador $f0.

2. Implementação:

```
.data
promptFibo:     .asciiz "O 30° número de Fibonacci é: "
promptPhi:      .asciiz "A razão áurea usando F_41 e F_40 é: "
newline:        .asciiz "\n"

.text
.globl main

# Função principal
main:
   # Calcular o 30° número de Fibonacci
   li $a0, 30          # n = 30
   jal fibonacci
   move $s1, $v0       # Armazenar resultado de fibonacci(30) em $s1

   # Calcular o 41° número de Fibonacci
   li $a0, 41          # n = 41
   jal fibonacci
   move $s2, $v0       # Armazenar resultado de fibonacci(41) em $s2

   # Calcular o 40° número de Fibonacci
   li $a0, 40          # n = 40
   jal fibonacci
   move $s3, $v0       # Armazenar resultado de fibonacci(40) em $s3

   # Calcular a razão áurea F_41 / F_40
   move $a0, $s2       # Numerador
   move $a1, $s3       # Denominador
   jal div_float
```

```
    # Imprimir o 30° número de Fibonacci
    la $a0, promptFibo    # Carregar mensagem do prompt
    li $v0, 4
    syscall

    move $a0, $s1        # Colocar resultado em $a0
    li $v0, 1            # Código de serviço para impressão de inteiro
    syscall

    la $a0, newline      # Imprimir nova linha
    li $v0, 4
    syscall

    # Imprimir a razão áurea
    la $a0, promptPhi     # Carregar mensagem do prompt
    li $v0, 4
    syscall

    mov.s $f12, $f0      # Colocar resultado em $f12
    li $v0, 2            # Código de serviço para impressão de float
    syscall

    # Finalizar programa
    li $v0, 10           # Código de serviço para sair do programa
    syscall

# Função para calcular o n-ésimo termo da sequência de Fibonacci
fibonacci:
    addi $sp, $sp, -8    # Ajustar pilha
    sw $ra, 4($sp)       # Salvar endereço de retorno
    sw $a0, 0($sp)       # Salvar argumento n

    li $t0, 0            # F(0) = 0
    li $t1, 1            # F(1) = 1

    bgt $a0, 1, fib_loop  # Se n > 1, ir para o loop
    beq $a0, 0, fib_exit  # Se n == 0, retornar 0
    move $v0, $t1        # Se n == 1, retornar 1
    j fib_return

fib_loop:
    li $t2, 2            # Iniciar contador
fib_next:
    add $t3, $t0, $t1    # c = a + b
    move $t0, $t1        # a = b
    move $t1, $t3        # b = c
    addi $t2, $t2, 1     # contador++
```

```
    ble $t2, $a0, fib_next# Se contador <= n, repetir

    move $v0, $t3       # Colocar resultado em $v0
    j fib_return

fib_exit:
    move $v0, $t0       # Retornar F(0)

fib_return:
    lw $ra, 4($sp)      # Restaurar endereço de retorno
    lw $a0, 0($sp)      # Restaurar argumento n
    addi $sp, $sp, 8    # Ajustar pilha
    jr $ra              # Retornar

# Função para divisão de inteiros, retorna float
div_float:
    mtc1 $a0, $f12      # Mover numerador para $f12
    mtc1 $a1, $f14      # Mover denominador para $f14
    cvt.s.w $f12, $f12  # Converter numerador para float
    cvt.s.w $f14, $f14  # Converter denominador para float
    div.s $f0, $f12, $f14 # Dividir $f12 por $f14
    jr $ra              # Retornar
```

Link do GitHub

https://github.com/douglaslima-ubec/assembly-fibonacci

3. Teste

Registradores inteiros

Registers | Coproc 1 | Coproc 0

Name	Number	Value
$zero	0	0
$at	1	268500992
$v0	2	10
$v1	3	0
$a0	4	268501022
$a1	5	102334155
$a2	6	0
$a3	7	0
$t0	8	63245986
$t1	9	102334155
$t2	10	41
$t3	11	102334155
$t4	12	0
$t5	13	0
$t6	14	0
$t7	15	0
$s0	16	0
$s1	17	832040
$s2	18	165580141
$s3	19	102334155
$s4	20	0
$s5	21	0
$s6	22	0
$s7	23	0
$t8	24	0
$t9	25	0
$k0	26	0
$k1	27	0
$gp	28	268468224
$sp	29	2147479548
$fp	30	0
$ra	31	4194352
pc		4194432
hi		0
lo		0

$t0: armazena o primeiro valor do cálculo da sequência de Fibonacci

$t1: armazena o segundo valor do cálculo da sequência de Fibonacci

$t3: armazena o resultado do cálculo da sequência de Fibonacci **Fn = F(n-1) + F(n-2)**

$s1: armazena o valor do elemento na posição 30

$s2: armazena o valor do elemento na posição 41

$s3: armazena o valor do elemento na posição 40

$sp: armazena o endereço de memória da pilha, conhecido como ponteiro da pilha

$ra: armazena o endereço de retorno das funções

Registradores de ponto flutuante

Registers | Coproc 1 | Coproc 0

Name	Float	Double
$f0	1.6180341	5.289158814E-315
$f1	0.0	
$f2	0.0	0.0
$f3	0.0	
$f4	0.0	0.0
$f5	0.0	
$f6	0.0	0.0
$f7	0.0	
$f8	0.0	0.0
$f9	0.0	
$f10	0.0	0.0
$f11	0.0	
$f12	1.6180341	5.289158814E-315
$f13	0.0	
$f14	1.02334152E8	6.36287474E-315
$f15	0.0	
$f16	0.0	0.0
$f17	0.0	
$f18	0.0	0.0
$f19	0.0	
$f20	0.0	0.0
$f21	0.0	
$f22	0.0	0.0
$f23	0.0	
$f24	0.0	0.0
$f25	0.0	
$f26	0.0	0.0
$f27	0.0	
$f28	0.0	0.0
$f29	0.0	
$f30	0.0	0.0
$f31	0.0	

$f0: armazena o valor da razão áurea de F(41) e F(40)

4. Conclusão

O trabalho foi desenvolvido com o objetivo de implementar o cálculo da sequência de Fibonacci e o cálculo da razão áurea dos elementos da sequência. Além disso, realizamos alguns testes com os registradores para validar os resultados, o que permitiu demonstrar a eficácia do programa em calcular a sequência de Fibonacci e a razão áurea.

Por meio deste trabalho, colocamos em prática conceitos importantes da arquitetura MIPS, como a criação de variáveis, manipulação de valores em registradores e memória, instruções aritméticas, instruções de desvio e funções.

Referências

SEVY, Jonathan. MIPS Architecture and Assembly Language. Jsevy. Disponível em: <https://jsevy.com/architecture/MIPSRef>. Acesso em: 14 de jun. de 2024.

Atenção:

1. *O texto deve ser formatado com a fonte Calibre, tamanho 12;*
2. *As formatações dos títulos e subtítulos devem ser mantidas;*
3. *O código-fonte aqui colado deve apresentar fundo branco, e espaçamento 0 pt antes e depois;*
4. *As partes deste documento devem ser mantidas;*
5. *Todo o texto escrito de vermelho diz respeito a instruções e deve ser retirado do documento de entrega.*
6. *O trabalho deverá ser entregue no formato PDF.*
7. *Caso o trabalho seja submetido mais de uma vez, será considerado o último documento enviado.*
8. *O nome e o sobrenome de cada aluno deve ser indicado no rodapé.*
9. *As notas serão disponibilizadas em área específica do AVA.*
10. *Antes de enviar seu trabalho, revise o texto escrito e atualize o índice.*