

UNIVERSIDADE DO VALE DO ITAJAÍ
CIÊNCIA DA COMPUTAÇÃO
ARQUITETURA E ORGANIZAÇÃO DE PROCESSADORES
PROFESSOR: THIAGO FELSKI PEREIRA

AUTORES:

GUSTAVO BARON LAURITZEN

MATHEUS BARON LAURITZEN

GABRIEL BÓSIO

AVALIAÇÃO 03
PROGRAMAÇÃO DE PROCEDIMENTOS

ITAJAÍ
18/05/2023

Problema 01:

- Código feito na linguagem de montagem:

```
1. # Disciplina: Arquitetura e Organização de Computadores
2. # Atividade: Avaliação 03 – Programação de Procedimentos
3. # Programa 01
4. # Grupo: - Gabriel Bosio
5. #           - Gustavo Baron Lauritzen
6. #           - Matheus Baron Lauritzen
7.
8. .data
9.
10. vetor:                                     .word
    0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33
    ,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,6
    3,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,
    93,94,95,96,97,98,99
11.
12. cout: .asciz "Valor da soma dos numeros do vetor: "
13. cout2: .asciz "Digite a posicao(2-100): "
14. .text
15.
16. jal zero, main
17.
18. soma_vetor:
19. loop1:
20. addi a1, a1, -1 #a1 = posicoes, subtrai para usar como indice
21.
22. slli t0, a1, 2 #calculo de enderecamento e carregamento do valor
23. add t1, t0, a0
24. lw s0, 0(t1)
25.
26. add s1, s1, s0 #soma = soma + s0*v[i]"
27.
28. bgt a1, zero, loop1
29.
30. addi sp, sp, -4 #guardando a soma na pilha
31. sw s1, 0(sp)
32.
33. jalr ra, 0
34.
35. main:
36. addi t5, zero, 2#utilizado como parametro de condicao
37. addi t6, zero, 100#utilizado como parametro de condicao
38. while1:
39.
40. addi a7, zero, 4 #impressao da descricao
41. la a0, cout2
42. ecall
43.
44. addi a7, zero, 5
45. ecall
46. add t4, zero, a0
47.
48. blt t4,t5,while1
```

```

49. bgt t4,t6,while1
50.
51. la a0, vetor
52. add a1, zero, t4
53. jal ra, soma_vetor #chamada de procedimento
54.
55. addi a7, zero, 4 #impressao da descricao
56. la a0, cout
57. ecall
58.
59. lw s1, 0(sp) #lendo a soma da pilha
60. addi sp, sp, 4
61.
62. addi a7, zero, 1#impressao do resultado soma
63. add a0, zero, s1
64. ecall

```

Explicação do código:

A função “soma_vetor” é responsável por realizar a soma dos valores do vetor. Ela possui um loop (loop1) que itera sobre as posições do vetor, iniciando com o valor fornecido em a1 (posições) e decrementando-o a cada iteração. A multiplicação “slli” é utilizada para calcular o deslocamento do endereço do vetor e obter o valor a ser somado (s0). A soma é acumulada em “s1”. O loop continua enquanto “a1” for maior que zero. Após o loop, o valor da soma é armazenado na pilha e a função retorna usando “jalr ra, 0”.

A função “main” é o ponto de entrada principal. Ela inicializa os valores dos registradores “t5” e “t6” com 2 e 100, respectivamente, que serão utilizados como parâmetros de condição do loop “while1”. O loop “while1” solicita ao usuário que digite uma posição (2-100) e armazena o valor fornecido em “t4”. Se o valor de “t4” estiver fora do intervalo desejado, o loop se repete até que um valor válido seja fornecido.

Em seguida, o endereço do vetor é carregado em “a0” e o valor da posição fornecida pelo usuário é carregado em “a1”. A função “soma_vetor” é chamada usando “jal ra, soma_vetor” para realizar a soma dos valores do vetor. Após o retorno da função “soma_vetor”, é realizada a impressão da descrição (cout) usando a “syscall” 4 (a7 é definido como 4 e o endereço da string é carregado em a0). Em seguida, o valor da soma é lido da pilha, o espaço da pilha é liberado e o resultado da soma é impresso na tela usando a “syscall 1”.

- **Capturas de tela:**

Entrada de dados:

Messages	Run I/O
<input type="button" value="Clear"/>	Digite a posicao(2-100): 100 Valor da soma dos numeros do vetor: 4950 -- program is finished running (dropped off bottom) --


Entrada da posição(int) que deve ser utilizada na soma dos valores do vetor.

Saída de dados:

Messages	Run I/O
<input type="button" value="Clear"/>	Digite a posicao(2-100): 100 Valor da soma dos numeros do vetor: 4950 -- program is finished running (dropped off bottom) --

Saída da soma dos valores do vetor até a posição passada pela Entrada de Dados.

Estatísticas de Instrução:


Instruction Statistics, Version 1.0 (Ingo Kofler)
×

Total:	<input type="text" value="628"/>	
ALU:	<input type="text" value="417"/>	<div><div></div>67%</div>
Jump:	<input type="text" value="3"/>	<div><div></div>1%</div>
Branch:	<input type="text" value="102"/>	<div><div></div>16%</div>
Memory:	<input type="text" value="102"/>	<div><div></div>16%</div>
Other:	<input type="text" value="4"/>	<div><div></div>1%</div>

Tool Control

Estatística de um vetor com 100 posições, onde foi passado como parâmetro a posição máxima do vetor(int 100) para fazer a soma dos seus valores.

Entrada de dados:

Messages	Run I/O
<div>Clear</div>	Digite a posicao(2-100): 50 Valor da soma dos numeros do vetor: 1225 -- program is finished running (dropped off bottom) --

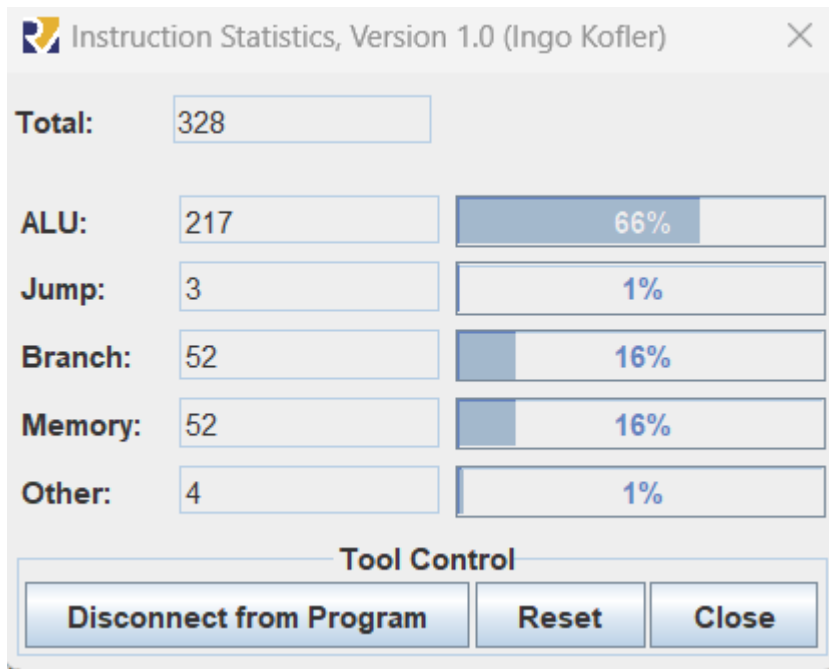
Entrada da posição(int) que deve ser utilizada na soma dos valores do vetor.

Saída de dados:

Messages	Run I/O
<div>Clear</div>	Digite a posicao(2-100): 50 Valor da soma dos numeros do vetor: 1225 -- program is finished running (dropped off bottom) --

Saída da soma dos valores do vetor até a posição passada pela Entrada de Dados

Estatísticas de Instrução:



Estatística de um vetor com 100 posições, onde foi passado como parâmetro a posição 50 do vetor(int 50) para fazer a soma dos seus valores.

Entrada de dados:

Messages	Run I/O
<div>Clear</div>	Digite a posicao(2-100): 15 Valor da soma dos numeros do vetor: 105 -- program is finished running (dropped off bottom) --

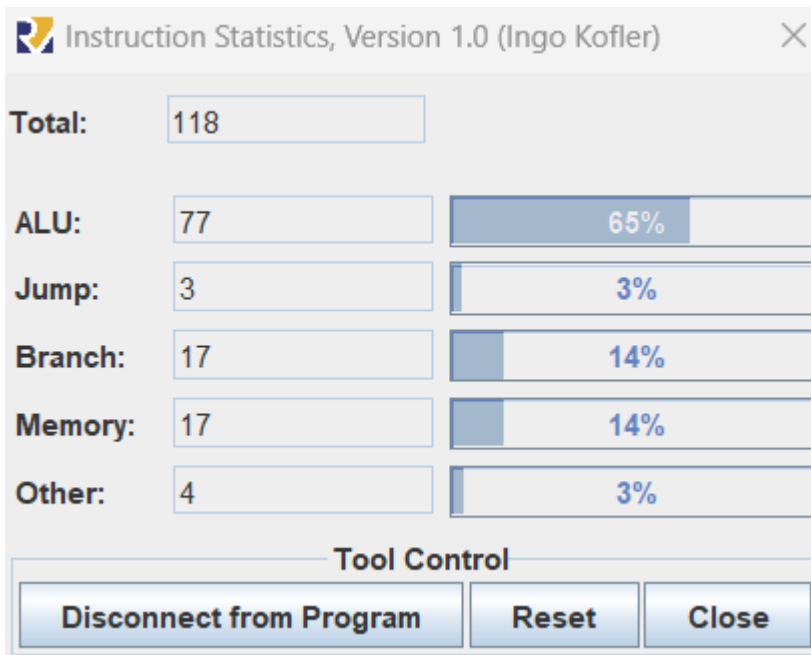
Entrada da posição(int) que deve ser utilizada na soma dos valores do vetor.

Saída de dados:

Messages	Run I/O
<div>Clear</div>	Digite a posicao(2-100): 15 Valor da soma dos numeros do vetor: 105 -- program is finished running (dropped off bottom) --

Saída da soma dos valores do vetor até a posição passada pela Entrada de Dados

- **Estatísticas de Instrução:**



Estatística de um vetor com 100 posições, onde foi passado como parâmetro a posição 15 do vetor(int 15) para fazer a soma dos seus valores.

Problema 02:

- **Código feito na linguagem de montagem:**

1. # Disciplina: Arquitetura e Organização de Computadores
2. # Atividade: Avaliação 03 – Programação de Procedimentos
3. # Programa 02
4. # Grupo: - Gabriel Bosio
5. # - Gustavo Baron Lauritzen
6. # - Matheus Baron Lauritzen
- 7.
8. .data
- 9.
10. vetor: .word

0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32,33

,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99
- 11.
12. cout: .asciz "Valor da soma dos numeros do vetor: "
13. cout2: .asciz "Digite a posicao(2-100): "
14. .text
- 15.
16. jal zero, main
- 17.
18. soma:
19. addi sp, sp, -8 #emplihlando s0 e ra no sp
20. sw s0, 4(sp)
21. sw ra, 0(sp)
- 22.

```

23. addi a1, a1, -1 #subtrai uma posicao
24.
25. bge a1, zero, recursao #checagem para a condicao de parada
26.
27. lw ra, 0(sp) #desempilha o endereco de retorno e guarda s0 no lugar
28. sw zero, 0(sp)
29.
30. jalr ra, 0
31.
32. recursao:
33. slli t0, a1, 2 #calcula de enderecamento e carregamento do valor
34. add t1, t0, a0
35. lw s0, 0(t1)
36.
37. jal ra, soma #repetir
38.
39. lw ra, 8(sp) #desempilha a soma dos anteriores, o proximo valor da pilha e o endereco de
    retorno
40. lw t0, 4(sp)
41. lw s0, 0(sp)
42.
43. addi sp, sp, 8 #diminui a pilha, deixando o espaco que pertencia o ra para guardar s0
44.
45. add s0, s0, t0 #soma o s0 com t0
46. sw s0, 0(sp)
47.
48. jalr ra, 0
49.
50. main:
51. addi t5, zero, 2#utilizado como parametro de condicao
52. addi t6, zero, 100#utilizado como parametro de condicao
53. while1:
54.
55. addi a7, zero, 4 #impressao da descricao
56. la a0, cout2
57. ecall
58.
59. addi a7, zero, 5
60. ecall
61. add t4, zero, a0
62.
63. blt t4, t5, while1
64. bgt t4, t6, while1
65.
66. la a0, vetor
67. add a1, zero, t4
68. addi s0, zero, 0
69.
70. jal ra, soma #chamada de procedimento
71.
72. lw s0, 0(sp)
73. addi sp, sp, 4
74.
75. addi a7, zero, 4 #impressao da descricao e da soma
76. la a0, cout
77. ecall
78.

```



```
79. addi a7, zero, 1
80. add a0, zero, s0
81. ecall
```

Explicação do código:

Na seção .data, é declarado o vetor de números inteiros chamado "vetor" com 100 elementos, variando de 0 a 99. Além disso, são declaradas duas strings (cadeias de caracteres) chamadas "cout" e "cout2". Na seção .text, o programa principal começa com o salto incondicional jal zero, main para o rótulo main, que é onde a execução principal do programa começa.

Em seguida, temos o procedimento soma. Antes de entrar no procedimento, são realizadas algumas instruções para salvar os registros necessários na pilha (stack). Isso é feito usando o registrador sp (ponteiro de pilha) e os registradores s0 e ra. Dentro do procedimento soma, é verificado se o valor do registrador a1 é maior ou igual a zero. Se for, a execução continua para o rótulo recursao. Caso contrário, ocorre o retorno da função, desempilhando o registrador ra e salvando o valor atual de s0. No rótulo recursao, o endereçamento do elemento do vetor é calculado com base no valor de a1 e a0. Em seguida, o valor é carregado do vetor e armazenado em s0. A instrução jal ra, soma é usada para chamar o procedimento soma novamente, passando o endereço atual em ra.

Após a chamada recursiva, os valores são desempilhados e restaurados. A soma atual s0 é adicionada ao valor armazenado em t0 e o resultado é armazenado em s0. O programa principal continua no rótulo main. É solicitado ao usuário que digite uma posição (2-100) usando a chamada de sistema para imprimir a string "Digite a posicao(2-100): ".

Em seguida, é feita uma verificação se o valor digitado (t4) está dentro do intervalo desejado (2-100). Se estiver fora desse intervalo, o programa retorna ao rótulo while1 para solicitar um novo valor. Se o valor estiver dentro do intervalo, o endereço do vetor é carregado em a0, o valor digitado é copiado para a1 e s0 é inicializado com zero. Em seguida, o procedimento soma é chamado para calcular a soma dos elementos do vetor a partir da posição especificada. O resultado da soma é armazenado em s0. Após o procedimento soma, o valor de s0 é recuperado da pilha e impresso na tela usando chamadas de sistema.

- **Capturas de tela:**

Entrada de dados:

Messages	Run I/O
<input type="button" value="Clear"/>	Digite a posicao(2-100): 100 Valor da soma dos numeros do vetor: 4950 -- program is finished running (dropped off bottom) --


Entrada da posição(int) que deve ser utilizada na soma dos valores do vetor.

Saída de dados:

Messages	Run I/O
<input type="button" value="Clear"/>	Digite a posicao(2-100): 100 Valor da soma dos numeros do vetor: 4950 -- program is finished running (dropped off bottom) --

Saída da soma dos valores do vetor até a posição passada pela Entrada de Dados

- **Estatísticas de Instrução:**

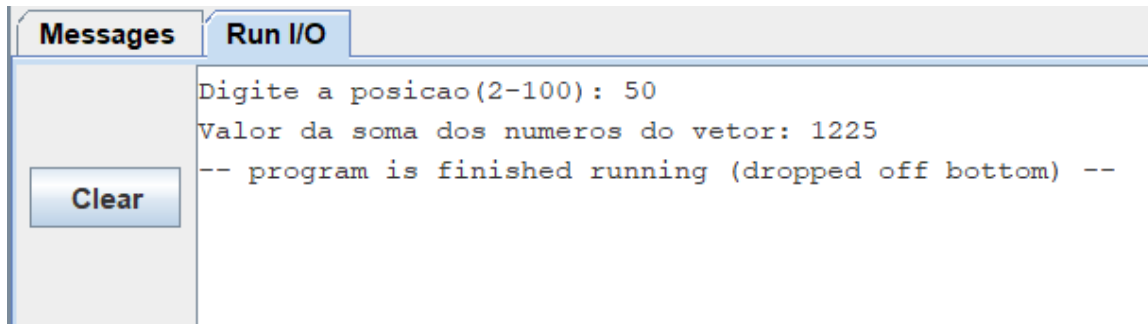

Instruction Statistics, Version 1.0 (Ingo Kofler)
×

Total:	<input type="text" value="1634"/>	
ALU:	<input type="text" value="619"/>	<div><div></div></div> 38%
Jump:	<input type="text" value="203"/>	<div><div></div></div> 12%
Branch:	<input type="text" value="103"/>	<div><div></div></div> 6%
Memory:	<input type="text" value="705"/>	<div><div></div></div> 43%
Other:	<input type="text" value="4"/>	<div><div></div></div> 0%

Tool Control

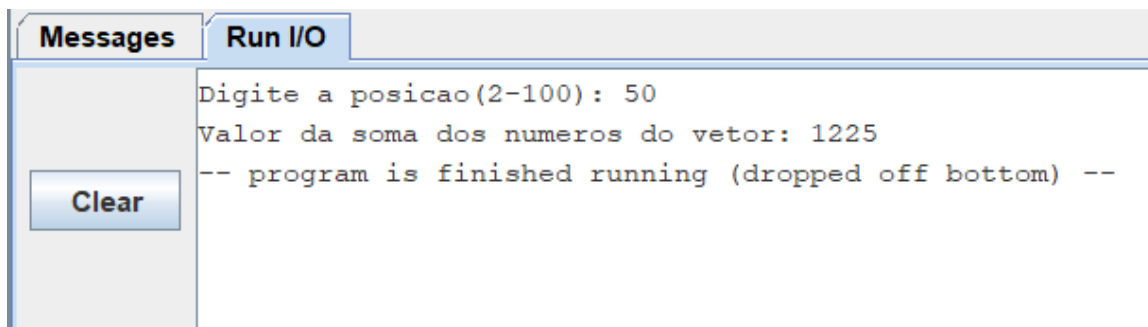
Estatística de um vetor com 100 posições, onde foi passado como parâmetro a posição máxima do vetor(int 100) para fazer a soma dos seus valores.

Entrada de dados:



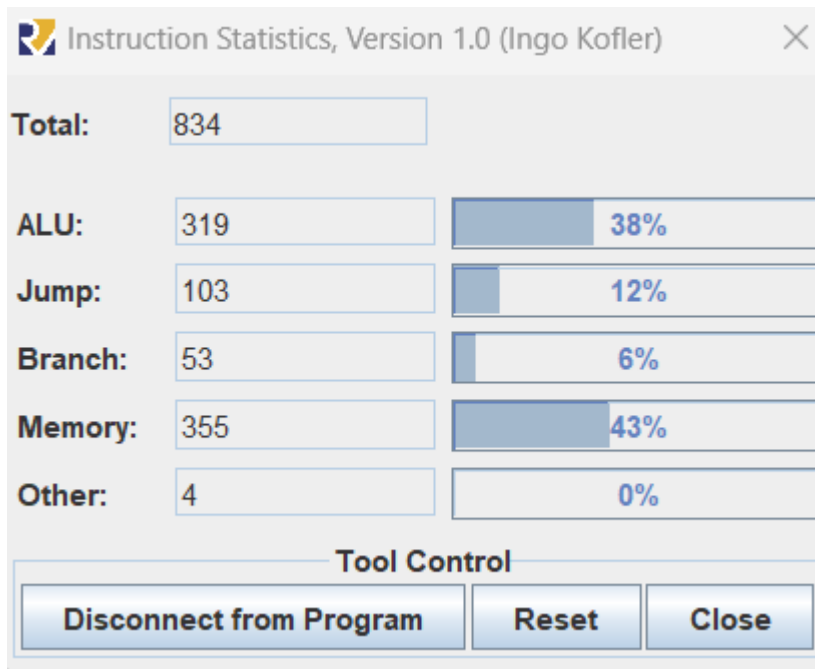
Entrada da posição(int) que deve ser utilizada na soma dos valores do vetor.

Saída de dados:



Saída da soma dos valores do vetor até a posição passada pela Entrada de Dados

- **Estatísticas de Instrução:**



Estatística de um vetor com 100 posições, onde foi passado como parâmetro a posição 50 do vetor(int 50) para fazer a soma dos seus valores.

Entrada de dados:

Messages	Run I/O
<div>Clear</div>	Digite a posicao(2-100): 15 Valor da soma dos numeros do vetor: 105 -- program is finished running (dropped off bottom) --

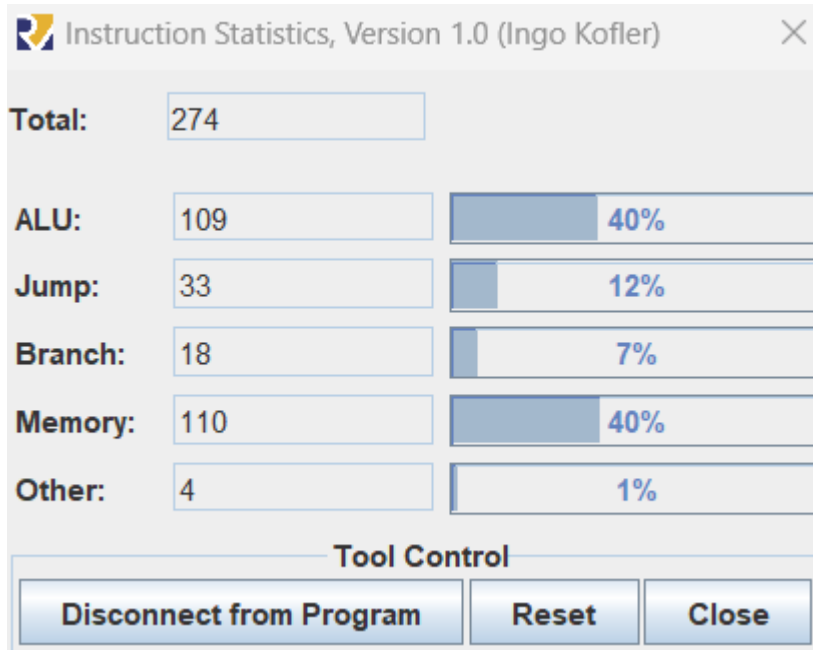
Entrada da posição(int) que deve ser utilizada na soma dos valores do vetor.

Saída de dados:

Messages	Run I/O
<div>Clear</div>	Digite a posicao(2-100): 15 Valor da soma dos numeros do vetor: 105 -- program is finished running (dropped off bottom) --

Saída da soma dos valores do vetor até a posição passada pela Entrada de Dados

- **Estatísticas de Instrução:**



Estatística de um vetor com 100 posições, onde foi passado como parâmetro a posição 15 do vetor(int 15) para fazer a soma dos seus valores.

Análise dos códigos:

Os dois códigos apresentados têm o objetivo de calcular a soma dos elementos de um vetor em uma determinada posição. O primeiro código utiliza uma abordagem recursiva, enquanto o segundo código utiliza um loop iterativo.

Em termos de desempenho, a solução iterativa tende a ser mais eficiente do que a solução recursiva, conforme visto nas estatísticas de instrução. Isso ocorre porque a solução iterativa evita a sobrecarga associada às chamadas recursivas, o que se torna mais evidente à medida que o tamanho do vetor aumenta. O código iterativo percorre o vetor uma vez, somando os elementos diretamente, enquanto o código recursivo faz chamadas recursivas para cada elemento do vetor, o que resulta em um número maior de instruções executadas.

Levando em consideração a quantidade de dados, se o vetor for pequeno, a diferença de desempenho entre as duas soluções pode não ser tão significativa. No entanto, à medida que o vetor aumenta de tamanho, o desempenho do código recursivo degrada mais rapidamente em comparação com o código iterativo. Isso ocorre porque a sobrecarga associada às chamadas recursivas aumenta exponencialmente com o tamanho do vetor, enquanto o código iterativo executa um número fixo de instruções, independentemente do tamanho do vetor.

Outrossim, em relação a abordagem recursiva ser melhor ou pior do que a iterativa, neste caso, a resposta é que a recursão não é a melhor opção. Embora a

recursão possa ser útil e elegante em alguns casos, quando se trata de percorrer um vetor para calcular uma soma, a abordagem iterativa é mais eficiente e evita o consumo excessivo de recursos.

Em resumo, a solução iterativa apresenta um melhor desempenho em relação à solução recursiva para o cálculo da soma dos elementos de um vetor. Conforme o tamanho do vetor aumenta, a diferença de desempenho entre as duas soluções se torna mais pronunciada, devido à sobrecarga associada às chamadas recursivas. Portanto, pode-se chegar à conclusão de que a abordagem iterativa é preferível neste caso.