RELATÓRIO TRADUÇÃO DE CÓDIGO EM C PARA ASSEMBLY DO MIPS E COMPARAÇÃO COM O CHAT-GPT

Caio Henrique Carvalho de Paiva

Pontifícia Universidade Católica de minas gerais (PUC-MG)

CEP 37714-620 - Poços de Caldas – MG – Brasil

Curso de Ciências da Computação

Laboratório de Arquitetura e Organização de Computadores

Caio.c.paiva07@gmail.com

Abstract. This article compares two implementations of a factorial function in assembly, highlighting the structural and functional differences between the original code and the code generated by ChatGPT. The analysis covers the order of functions, stack management, use of registers, and the recursive call logic. The article concludes that while both approaches attempt to solve the factorial calculation problem, the ChatGPT implementation contains significant errors that prevent its correct functioning, particularly in the function order and register handling during recursion.

Resumo. Este artigo compara duas implementações de uma função fatorial em assembly, destacando as diferenças estruturais e funcionais entre o código original e o código gerado pelo ChatGPT. A análise aborda a ordem das funções, a gestão da pilha, o uso de registradores e a lógica de chamada recursiva. O artigo conclui que, embora ambas as abordagens tentem resolver o problema do cálculo fatorial, a implementação do ChatGPT possui erros significativos que impedem seu funcionamento correto, especialmente na ordem das funções e no manuseio de registradores durante a recursão.

Diferenças entre meu código e o código provido pelo ChatGPT,

1. Ordem: das funções main, fatorial e retorno.

Enquanto eu começo com a função main chamando a função fatorial, o código apresentado pelo chat realiza o inverso e faz a função fatorial antes da main, acredito que o meu jeito seja mais adequado, porém não existe problema em começar com a função fatorial.

2. Função main:

Minha função main:

```
1     .data
2  output_message: .asciiz "O fatorial de 10 é:"
3     .text
4     .globl main
5
6  main:
7     # Chamada da função fatorial(10)
8     li $a0, 10
9     jal fatorial                # Coloca o valor 10 em N e chama a função, o retorno da função é colocado em $t0
10    move   $t0, $v0
11
12    # Impressão do resultado
13    li $v0, 4
14    la $a0, output_message      # Imprime a mensagem pré programada
15    syscall
16
17    li $v0, 1
18    move $a0, $t0               # Imprime o resultado da função
19    syscall
20
21    # Encerrar programa
22    li $v0, 10
23    syscall
```

Função main do ChatGPT:

```
34 main:
35     # Chama a função fatorial com o argumento 10
36     li $a0, 10              # Argumento para fatorial
37     jal fatorial
38
39     # Imprime o resultado
40     move $a0, $v0           # Move o resultado para o registrador de argumento
41     li $v0, 1               # Código de syscall para imprimir inteiro
42     syscall
43
44     # Finaliza o programa
45     li $v0, 10              # Código de syscall para sair
46     syscall
```

Vemos que o chat não fez uma string base para o resultado do fatorial, o modo como ele passa o argumento para a função é o mesmo, utilizando li para passar 10 como argumento, |linha 8 meu código| |linha 36 Código do ChatGPT| Modo de imprimir o resultado é o mesmo.

3. Função fatorial:

Minha função fatorial:

```
25  fatorial:
26      # Prologue
27      addi $sp, $sp, -8           # Faz espaço na pilha para salvar $ra e $s0
28      sw $ra, 4($sp)              # Salva $ra na pilha
29      sw $s0, 0($sp)              # Salva $s0 na pilha
30      move $s0, $a0               # Salva N em $s0
31
32      # Corpo da função
33      li $v0, 1                   # se N <= 1, retorna 1
34      ble $s0, $v0, return_value
35
36      # Chamada recursiva: fatorial(n-1)
37      addi $a0, $s0, -1           # N - 1
38      jal fatorial                # Chamada recursiva
39      move $t0, $v0               # Move o resultado (em $v0) para $t0
40
41      # Cálculo do fatorial:
42      mul $v0, $s0, $t0
43
```

Função fatorial do ChatGPT:

```
1   .data
2   output_msg: .asciiz "%d\n"
3
4   .text
5   .globl main
6
7   # Função para calcular o fatorial
8   fatorial:
9       # Prologue
10      addi $sp, $sp, -4       # Reserva espaço na pilha
11      sw $ra, 0($sp)          # Salva o endereço de retorno
12
13      # Verifica se n <= 1
14      ble $a0, 1, base_case
15
16      # Caso n > 1, calcula n * fatorial(n-1)
17      addi $sp, $sp, -4       # Reserva espaço na pilha para o argumento
18      sw $a0, 0($sp)          # Salva o valor de n na pilha
19      addi $a0, $a0, -1       # Calcula n-1
20      jal fatorial            # Chama recursivamente a função fatorial
21      lw $a0, 0($sp)          # Restaura o valor de n da pilha
22      lw $t1, 0($sp)          # Carrega o resultado da chamada recursiva
23      mul $v0, $a0, $t1       # Calcula n * fatorial(n-1)
24
25      # Epilogue
26      addi $sp, $sp, 4        # Libera espaço na pilha
27      lw $ra, 0($sp)          # Restaura o endereço de retorno
28      jr $ra                  # Retorna
29
30  base_case:
31      li $v0, 1               # Retorna 1
32      jr $ra                  # Retorna
33
```

Vemos que minha função fatorial começa reservando espaço na pilha de forma diferente à do ChatGPT, onde o chat salva apenas $ra, portando utiliza –4 para reservar espaço na pilha, enquanto eu utilizo –8 para poder salvar tanto $ra como $s0, isto é essencial para o exercício em questão.

Após isto salvo N em $s0 para que a função funcione corretamente e utilizo $v0 com o valor 1 para realizar a comparação com a ble, aonde se N for <= 1 deverá ir para a função de retorno, o chat realiza operação parecida, porém manda para a função base_case. Se N passar pela ble, realizo uma operação para decrescer N em 1 e já chamo a função fatorial de forma recursiva, realizando as operações de reserva de espaço em

pilha, salvamento da variável N, comparação com a ble novamente e decrescimento de N em –1, realizo essas operações até que ble = true.

Já o ChatGPT realiza a operação de reserva de pilha incorretamente, decresce N e chama fatorial novamente até ble = true

Quando isto acontece o ChatGPT carrega em $v0 o valor 1 e retorna à função chamadora, carregando valores iguais em $a0 e $t1, utilizando lw de forma incorreta já que salvou apenas um dado. Continua neste looping até o jr ra retornar à função chamadora no main e printa o resultado.

Após isso, na minha implementação temos a função return_value:

```
44  return_value:
45      lw $ra, 4($sp)          # Restaura $ra da pilha
46      lw $s0, 0($sp)          # Restaura $s0 da pilha
47      addi $sp, $sp, 8        # Desaloca espaço da pilha
48      jr $ra                  # Retorna para a função chamadora
```

Está função serva para voltarmos restaurando $s0 e $ra, deslocando espaço e utilizando o jr ra para voltarmos a linha 38 do meu código, onde abaixo dela moveremos para $t0 o resultado e o multiplicaremos pelo N, fazendo assim o fatorial e entrando em return_value até que o jr ra retorne para a chamada da função no main, dando continuidade para o output do resultado.

O código do ChatGPT não funciona, possui alguns erros e um dos principais é a ordem em que o código está escrito, onde o código já começa em fatorial, fazendo com que a main nem seja executada e portando nunca foi passado um valor para N, fazendo com que a primeira execução da ble seja true e já mandará executar a linha 32, onde existe um jr ra, que é impossível de ser executado pois não existiu nenhum jal.

Mesmo corrigindo este erro, o código continua a não funcionar pois possui mais erros na chamada de jr ra na linha 40, onde dá erro.