Análise de uma função recursiva em Assembly MIPS - Humanos e ChatGPT

Henrique Luiz de Almeida Lopes

Pontifícia Universidade Católica de Minas Gerais (PUC-MG)
Av. Padre Cletus Francis Cox, 1661 - Jardim Country Club, Poços de Caldas - MG

Henriquelopes28@outlook.com

1. Proposta

Atualmente muito tem sido abordado a respeito de IA's gerando códigos, a proposta realizada visa confrontar a confiabilidade desses códigos. Foi proposto a criação de um algoritmo em Assembly MIPS que fizesse de forma recursiva o fatorial de 10, após isso foi proposto o mesmo para o ChatGPT - IA mais popular na comunidade.

Uma primeira análise foi realizada, pôde-se perceber que o código gerado pelo ChatGPT não funciona, o que será explorado a frente.

2. Código não didático

Antes de uma análise mais profunda sobre o código do chatGPT, foi percebido a falta de legibilidade do código, apesar das linhas comentadas explicarem uma linha de raciocínio, a compreensão é difícil e muitas vezes sem sentido (como pode-se observar na figura 1).

```
7  # Função fatorial
8  fatorial:
9      # Prologue
10     addi $sp, $sp, -8      # Reserva espaço na pilha
11     sw $ra, 4($sp)         # Salva o endereço de retorno
12     sw $a0, 0($sp)         # Salva o valor de n
13
14     # Corpo da função
15     lw $t0, 0($a0)         # Carrega n
16     li $t1, 1              # Carrega 1
17     ble $t0, $t1, base     # Verifica se n <= 1
18     addi $a0, $a0, -1      # n - 1
19     jal fatorial           # Chama fatorial(n - 1)
20     lw $t2, 0($a0)         # Carrega n de volta
21     mul $v0, $t2, $v0      # Multiplica n pelo resultado recursivo
22     j end
```

Figura 1. Função recursiva Fatorial - ChatGPT

3. Análise do código

A função gerada pela IA, cria o erro “address out of range 0x00000000” na linha 15, o erro foi cometido por tratar de maneira incorreta uma pilha em Assembly MIPS, o que provavelmente esgota a memória, tenta resgatar um valor em uma posição inexistente na mesma ou cria um loop infinito.

Em contrapartida, uma abordagem correta e concisa dessa função é observado na Figura 2, onde nenhum erro de memória ocorre e consegue tratar da pilha de forma correta, salvando os valores e depois multiplicando cada um no retorno de cada chamada.

```
.globl fatorial
fatorial:

subu $sp, $sp, 8
sw $ra, ($sp)
sw $s0, 4($sp)

# testa se chegou em 0
li $v0, 1 # retorna 1
beq $a0, 0, fim_fatorial

# chama o fatorial-1
move $s0, $a0
sub $a0, $a0, 1
jal fatorial

# multiplica nos retornos da funcao
mul $v0, $s0, $v0

fim_fatorial:

# volta os valores anteriores da stack
lw $ra, ($sp)
lw $s0, 4($sp)
addu $sp, $sp, 8

jr $ra
```

Figura 2. Função recursiva Fatorial - Humano

4. Conclusão

Ao observar as duas Figuras é nítido os perigos de confiar cegamente em códigos Assembly MIPS gerados por IA, além de praticamente sempre não serem didáticos e até terem uma performance abaixo de códigos feitos por humanos, ao usar linguagens de baixo nível, o tratamento da memória deve ser pontual e sem erros, o que claramente o ChatGPT não realiza com frequência.