

Trabalho I - Assembly do Processador MIPS

Implementação da Função de Ackermann Recursiva – V.2

Este documento descreve o primeiro trabalho da disciplina de Organização e Arquitetura de Processadores. O trabalho consiste na compreensão de um problema algorítmico, descrição deste problema em linguagem de alto nível, compreensão da ISA do MIPS e mapeamento do algoritmo no assembly do MIPS considerando todo o ISA (*Instruction Set Architecture*).

A atividade algorítmica envolve trabalhar com técnicas e fundamentos apreendidos na disciplina de OAP; dentre estes estão (i) a programação em linguagem de máquina do MIPS, (ii) a implementação algorítmica com o uso de função e, possivelmente macros, (iii) o salvamento e recuperação de registradores em pilha ao trabalhar com funções, (iv) o emprego de recursividade e (v) o interfaceamento do programa com o sistema operacional.

A atividade, que deverá ser realizada em **grupos de até 4 alunos**, envolve também o emprego do ambiente MARS para descrição e verificação do comportamento do algoritmo, e uma documentação adequada que apresente o desenvolvimento de todas as atividades requisitadas.

1. Especificação Técnica do Trabalho

O grupo deve implementar uma versão específica da função de Wilhelm Ackermann, mais conhecida como função de Ackermann. Na teoria da recursão, Ackermann é um exemplo de uma função computável total que não é recursiva primitiva. Todas as funções recursivas primitivas são totais e computáveis, mas a função de Ackermann ilustra que nem todas as funções computáveis totais são recursivas primitivas. Uma das inúmeras variantes da função original é a função Ackermann-Péter $A(m, n)$ de dois argumentos inteiros m e n não negativos, que é definida por:

$$A(m, n) = \begin{cases} n + 1 & \text{se } m = 0 \\ A(m - 1, 1) & \text{se } m > 0 \text{ e } n = 0 \\ A(m - 1, A(m, n - 1)) & \text{se } m > 0 \text{ e } n > 0 \end{cases}$$

Esta regra de recursividade faz a função de Ackermann crescer muito rapidamente, mesmo para pequenos valores de m e n . A seguir temos cinco exemplos da aplicação da função de Ackermann, ilustrando os passos da recursividade.

$$\begin{aligned} 1) \quad A(0, 1) &= 1+1 \\ &= 2 \end{aligned}$$

$$\begin{aligned} 2) \quad A(1, 0) &= A(1-1, 1) \\ &= A(0, 1) \\ &= 1+1 \\ &= 2 \end{aligned}$$

$$\begin{aligned} 3) \quad A(1, 1) &= A(1-1, A(1, 1-1)) \\ &= A(0, A(1, 0)) \\ &= A(0, A(1-1, 1)) \\ &= A(0, A(0, 1)) \\ &= A(0, 1+1) \\ &= A(0, 2) \\ &= 2+1 \\ &= 3 \end{aligned}$$

$$\begin{aligned}
4) \quad A(1,2) &= A(1-1, A(1,2-1)) \\
&= A(0, A(1, 1)) \\
&= A(0, A(1-1, A(1,1-1))) \\
&= A(0, A(0, A(1, 0))) \\
&= A(0, A(0, A(1-1, 1))) \\
&= A(0, A(0, A(0, 1))) \\
&= A(0, A(0, 1+1)) \\
&= A(0, A(0, 2)) \\
&= A(0, 2+1) \\
&= A(0, 3) \\
&= 3+1 \\
&= 4
\end{aligned}$$

$$\begin{aligned}
5) \quad A(2,1) &= A(2-1, A(2,1-1)) \\
&= A(1, A(2, 0)) \\
&= A(1, A(2-1,1)) \\
&= A(1, A(1,1)) \\
&= A(1, A(1-1, A(1,1-1))) \\
&= A(1, A(0, A(1, 0))) \\
&= A(1, A(0, A(1-1,1))) \\
&= A(1, A(0, A(0,1))) \\
&= A(1, A(0, 1+1)) \\
&= A(1, A(0, 2)) \\
&= A(1, 2+1) \\
&= A(1, 3) \\
&= A(1-1, A(1, 3-1)) \\
&= A(0, A(1, 2)) \\
&= A(0, A(1-1, A(1,2-1))) \\
&= A(0, A(0, A(1-1, A(1,1-1)))) \\
&= A(0, A(0, A(0, A(1,0)))) \\
&= A(0, A(0, A(0, 1+1))) \\
&= A(0, A(0, A(0, 2))) \\
&= A(0, A(0, 2+1)) \\
&= A(0, A(0, 3)) \\
&= A(0, 3+1) \\
&= A(0, 4) \\
&= 4+1 \\
&= 5
\end{aligned}$$

A tabela a seguir apresenta valores de Ackermann para as combinações de $0 \leq m \leq 4$ e $0 \leq n \leq 3$ (Fonte: https://pt.wikipedia.org/wiki/Fun%C3%A7%C3%A3o_de_Ackermann).

Valores de $A(m, n)$					
$m \backslash n$	0	1	2	3	4
0	1	2	3	4	5
1	2	3	4	5	6
2	3	5	7	9	11
3	5	13	29	61	125

1.1. Detalhamento da Especificação

O programa deve apresentar o valor da função de Ackermann para um par de inteiros (m, n) lidos da entrada padrão, exibindo na saída padrão o resultado da função.

O programa deve ser implementado com funções, tendo pelo menos as duas funções

descritas a seguir. Contudo, o grupo pode implementar outras funções e macros, de forma a tornar o programa mais modular e legível.

- (i) Uma função recursiva para cálculo do valor de Ackermann em relação a m e n ;
- (ii) Uma função principal (main).

1.2. Detalhamento da Interface com o Usuário

O programa a ser entregue deve conter as seguintes funcionalidades:

- 1) Iniciar a execução apresentando a seguinte mensagem:

“Programa Ackermann”

“Componentes: <Lista de nomes dos alunos>”

- 2) Entrar em um laço de execução que somente termina quando for pressionado **um número negativo**.

“Digite os parâmetros m e n para calcular $A(m, n)$ ou -1 para abortar a execução”

- (i) Caso o usuário digitar **um número negativo**, seja para m ou para n , o programa encerra.
- (ii) Caso o usuário não digitar um número negativo:
 - a. O programa deve capturar dois inteiros e calcular a função recursiva.
 - b. Ao terminar o cálculo da função, o programa deve retornar o resultado em um formato similar ao descrito a seguir:

“ $A(2, 1) = 5$ ”

- (iii) Retornar para executar um novo laço.

2. Entregas

As principais atividades a serem realizadas e comprovadas através de uma documentação adequada estão descritas a seguir:

- 1) Algoritmo descrevendo o programa de alto nível (linguagem Java, C, português estruturado, ...). O programa deve conter as funções especificadas na descrição, considerando a recursividade requisitada;
- 2) Descrição em linguagem assembly do MIPS equivalente ao programa de alto nível descrito em “1”;
- 3) Captura de telas do MARS mostrando:
 - a. A área de código compilada;
 - b. O estado dos registradores ao término de uma execução;
 - c. A área de pilha utilizada para a recursividade; e
 - d. Um exemplo de execução do programa.

O grupo deve entregar as atividades descritas acima em um arquivo compactado contendo uma documentação em formato pdf. Adicionalmente, o mesmo programa assembly que deve estar no documento, também deve ser colocado dentro do arquivo compactado para possível verificar seu funcionamento no ambiente MARS.