Trabalho da Disciplina - Assembly do Processador MIPS

Implementação da Função de Ackermann Recursiva – V.4

Este documento descreve o trabalho da disciplina de Organização e Arquitetura de Processadores que consiste na compreensão de um problema algorítmico, descrição deste algoritmo em linguagem de alto nível, compreensão do ISA (*Instruction Set Architecture*) do MIPS e implementação do algoritmo em linguagem de montagem (assembly).

A atividade algorítmica envolve trabalhar com técnicas e fundamentos apreendidos na disciplina; dentre estes estão (i) a programação em linguagem de montagem, (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 3 alunos***, envolve 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. Abaixo segue uma variante da função Ackermann, que é definida por:

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

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.

```
1)  A(0,1) = 1+1
           = 2
```

```
2)  A(1,0) = A(1-1,1)+1
           = A(0,1)+1
           = 2+1
           = 3
```

```
3)  A(1,1) = A(1-1, A(1,1-1))
           = A(0, A(1, 0))
           = A(0, A(1-1,1)+1)
           = A(0, A(0,1)+1)
           = A(0, 2+1)
           = A(0, 3)
           = 3+1
           = 4
```

```
4)  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)+1))
           = A(0, A(0, A(0, 1)+1))
           = A(0, A(0, 2+1))
           = A(0, A(0, 3))
           = A(0, 3+1)
           = A(0, 4)
           = 4+1
           = 5
```

```
5)  A(2,1) = A(2-1, A(2,1-1))
           = A(1, A(2, 0))
           = A(1, A(2-1, 1)+1)
           = A(1, A(1, 1)+1)
           = A(1, A(1-1, A(1,1-1))+1)
           = A(1, A(0, A(1, 0)+1))
           = A(1, A(0, A(1-1,1)+1+1))
           = A(1, A(0, A(0,1)+2))
           = A(1, A(0, 1+1+2))
           = A(1, A(0, 4))
           = A(1, 5)
           = A(1-1, A(1, 5-1))
           = A(0, A(1, 4))
           ...
           = 8
```

A tabela a seguir apresenta valores para a variante proposta da função de Ackermann, i.e., A(m,n):

m\n	0	1	2	3	4
0	1	2	3	4	5
1	3	4	5	6	7
2	5	8	11	14	17
3	9	32	101	308	929

1.1. Detalhamento da Especificação

O programa deve apresentar o valor da variante 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 e duas macros quaisquer. 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:

"Variante da Função de Ackermann – <Data no formato dia/mês/ano>"

"Autores: <Lista de nomes dos alunos>"

2) Entrar em um laço de execução que somente termina quando for pressionado ***um número negativo***, seja para **n** ou para **m**.

"Digite os parâmetros m e n para calcular A(m, n) ou número negativo para abortar a execução"

(i) Caso o usuário digitar ***um número negativo***, *seja para* ***m*** *ou para* ***n****, o programa encerra.*

(ii) Caso contrário:

a. O programa deve ler dois inteiros da entrada padrão e calcular a função recursiva.

b. Ao terminar o cálculo da função, o programa deve exibir o resultado em um formato similar ao descrito a seguir:

"A(2, 1) = 5"

(iii) Retornar para executar um novo laço que permita novas entradas m e n.

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, C++, português estruturado, ...). O programa deve conter as funções especificadas na descrição e macros, 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 ao término do programa; 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 ser possível verificar seu funcionamento no ambiente MARS.